



Fakultät für Elektrotechnik, Informatik und Mathematik
Fakultät für Kulturwissenschaften

Diplomarbeit

Untersuchung, Konzeption und Realisierung eines Flex-basierten Reportingclients mit PHP-Backend

Moritz Christian
Matrikelnummer: 6361190
E-Mail: moritzchr@gmail.com

Paderborn, den 14. Mai 2009

vorgelegt bei
Prof. Dr. Gerhard Szwillus
und
Martin Körner

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Paderborn, den 14. Mai 2009

Moritz Christian

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation 2

1.2 Die Entwicklung des World Wide Web (WWW) 3

2 Grundlagen **7**

2.1 PHP: Hypertext Preprocessor 7

2.2 Model-View-Controller (MVC) 7

2.3 Zend Framework 8

2.4 Adobe Flex 8

2.4.1 MXML 10

2.4.2 ActionScript 10

2.4.3 Flex-Builder 11

2.4.4 Flex-Charting 11

2.5 E-Mail 12

2.5.1 Simple Mail Transfer Protocol (SMTP) 12

2.5.2 Kopfzeilen (Header) 13

2.5.3 Inhalt (Body) 13

2.6 Reporting 13

2.6.1 Mailing 14

2.6.2 Tracking 14

2.6.3 Bounces 16

2.6.4 Status 16

2.7 Data-Warehouse 17

3 Untersuchung **19**

3.1 Eignung von Flex 19

3.1.1 Vorteile 19

3.1.2 Nachteile 21

3.2 Alternativen zu Flex 21

3.2.1 Silverlight 22

3.2.2 JavaFX 23

3.2.3 Fazit 24

3.3 Kommunikation zwischen Flex und PHP 26

3.3.1 Webservice 27

3.3.2 Remoting und RPC 27

3.3.3 Serialisierung und Deserialisierung 28
3.3.4 Action Message Format (AMF) 28
3.3.5 Representational State Transfer (REST) 29
3.4 Schnittstellenvergleich 30
3.4.1 HTTPService 30
3.4.2 Simple Object Access Protocol (SOAP) 31
3.4.3 AMFPHP 32
3.4.4 Zend_Amf 32
3.4.5 Vergleichsanalyse 33
3.5 Fazit 35

4 Konzeption 37

4.1 Grundfunktionen des Reportings 38
4.1.1 Darstellung von Mailings 38
4.1.2 Filterung der Mailings 39
4.1.3 Status der Versendung 40
4.1.4 Klicks pro Stunde 42
4.1.5 Linkauswertung 42
4.1.6 Diagramme anzeigen und vergleichen 43
4.1.7 Mehrsprachigkeit 44
4.2 Geo-Tracking 45
4.3 MVC-Implementierung 46
4.4 Softwarearchitektur 46
4.4.1 Unified Modelling Language (UML) 46
4.4.2 Strukturierung 47
4.5 Verbindungsaufbau zum PHP-Backend 49
4.5.1 Ablauf 49
4.5.2 Remote-Objekt 49
4.6 Sicherheit 52
4.6.1 Sicherheit ohne SSL 53
4.6.2 Authentifizierung 53
4.6.3 Sessions 54
4.7 Darstellung 55
4.7.1 Anordnung der Hauptelemente 55
4.7.2 Dashboard 56
4.7.3 Barrierearme Flash-Applikationen 58
4.8 Usability Engineering 59
4.8.1 Ergonomische Gestaltung 59
4.8.2 Heuristiken 61
4.8.3 Usability in Rich Internet Applications 63
4.9 Empirische Evaluierung 63
4.9.1 Prototyp 63
4.9.2 Vorbereitung und Ablauf 64
4.9.3 Aufgabenstellung 65

4.9.4 Fazit und Analyse 66

5 Realisierung 69
5.1 Programmierung 69
5.1.1 Stylesheets 69
5.1.2 Ladezeiten 69
5.1.3 States und Transitions 70
5.2 Darstellung 70
5.2.1 Menü 71
5.2.2 Filter 72
5.2.3 Mailings 72
5.2.4 Dashboard 73
5.2.5 Reportingfunktionen 74
5.3 Pitfalls 77
5.3.1 Mehrsprachigkeit 78
5.3.2 Sessions 78
5.3.3 Remote-Objekte 78

6 Zusammenfassung und Ausblick 79

Anhang 81
A eLettershop Oberfläche 81
B CD Inhaltsverzeichnis 83

Abbildungsverzeichnis 85

Quellcodeverzeichnis 87

Literaturverzeichnis 89

1 Einleitung

Die vorliegende Arbeit findet in Kooperation mit arvato systems, dem IT-Dienstleister der Bertelsmann AG, statt. Dabei bedient die Abteilung NMI-TA den Markt für kommerziellen Nachrichtenversand, hauptsächlich E-Mail-Verkehr. Das derzeitige Kernprojekt von NMI-TA ist der eLettershop[1]. Er wurde ursprünglich intern genutzt, um Massen-E-Mails im Kundenauftrag zu verschicken. Aufgrund der großen Nachfrage wurde das Portal für die direkte Kundennutzung geöffnet. Im Zuge dessen erfolgte die Entwicklung eines benutzerfreundlichen Webinterfaces. Dieses erlaubt den Kunden das eigenständige Erstellen und Versenden von E-Mails an große Empfängerlisten. Diese Funktionalität ist bereits mittels gängiger Webtechnologien zufriedenstellend umgesetzt. Des Weiteren lassen sich die Reaktionen der E-Mail-Empfänger in Form eines Reportings darstellen, welches höhere Anforderungen an die Visualisierung und Benutzerinteraktion stellt. Aus der Kundenresonanz lässt sich schließen, dass die derzeitige technische Umsetzung der Reportingfunktionen unzureichend ist. Die wesentlichen Kritikpunkte betreffen die Darstellungsgeschwindigkeit sowie die fehlenden Varianten der Darstellungsmöglichkeiten. Das in dieser Arbeit betrachtete Projekt hat daher zum Ziel, eine intuitiv zu bedienende Applikation mit modernem Design zu erstellen, die dem Kunden ein performantes und individuell anpassbares Reporting bietet.

Das vorliegende Dokument soll jenen Entwicklern als Erfahrungsbericht dienen, die Flex und PHP kombinieren wollen. Die Reportingapplikation stellt ein praxisnahes Fallbeispiel dar. Es erlaubt anhand diverser Einzelbeispiele die konkrete technische Umsetzung mit Hilfe der Flex-Technologie. Begleitend dazu werden die Vor- und Nachteile sowie Alternativen diskutiert. Die Intention dieser Arbeit ist es, dem Entwickler die Technologie derart vorzustellen, dass er die geschilderten Erfahrungen auf eigene Projekte übertragen kann.

Die Motivation, die diesem Projekt zugrunde liegt, ist die Weiterentwicklung der Reportingfunktionalität. Im nachfolgenden Abschnitt soll zudem die allgemeiner gefasste Motivation aus Kundensicht, nämlich die Bedeutung des Reportings, erläutert werden.

[1] Weitere Informationen zum eLettershop hält das auf der CD befindliche Benutzerhandbuch bereit.

1.1 Motivation

Das Internet bietet in Form des E-Mail-Versands eine Ergänzung zu den klassischen Instrumenten der Marktanalyse (z. B. Telefonbefragungen). Dieser ermöglicht zudem, das Verhalten der Marktteilnehmer zu beeinflussen. Durch Beobachtbarkeit und Steuerbarkeit sind die notwendigen Voraussetzungen für eine Regelung[2] (vgl. Abb. 1.1) gegeben. Mit ihrer Hilfe lässt sich die Wirkung von E-Mail-Kampagnen (im Folgenden Mailings (siehe Abs. 2.6.1) genannt) optimieren. Der Markt stellt

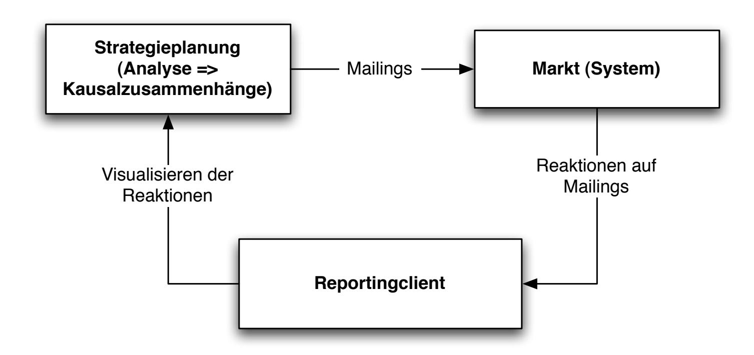


Abbildung 1.1: Reporting im Regelkreis

hierbei das zu regelnde und zu beobachtene System dar, welches durch Mailings angeregt wird. Die darauf erfolgenden Reaktionen der Empfänger werden im Reportingclient in der Art visualisiert, dass sich die anschließende Analyse vereinfacht. Ihr Ziel ist das Ermitteln von Kausalzusammenhängen. Dieses Wissen wird bei der Strategieplanung verwendet, um die Parameter zukünftiger Mailings (wie z. B. Form der Anrede, Gestaltung des Inhalts etc.) anzupassen. Die Rückkopplung ergibt sich dadurch, dass die so modifizierten Mailings verschickt und die Reaktionen darauf ebenfalls ausgewertet werden. Neben der Optimierung der eigentlichen Mailings kann die Beobachtungsfunktion als Grundlage für weitere Marketingstrategien dienen. Ihre Maßnahmen überspringen dabei die Grenzen des Mediums E-Mail. Dieses soll anhand des folgenden Beispiels eines Discounters verdeutlicht werden.
Es soll die Standortfrage für neue Filialen beantwortet werden. Hierzu wird mit Hilfe des Geo-Trackings (siehe Abs. 4.2) die geographische Verteilung der E-Mail-Empfänger und somit potentieller Kunden ermittelt. Treten dabei Häufungspunkte auf, an denen zudem keine Niederlassungen existieren, so ist ein potentiell günstig gelegener Standort gefunden.
Grundsätzlich ist bei der Analyse von Mailings die Zusammensetzung der Empfängergruppe zu beachten. Sie entspricht in der Regel nicht exakt der Gesellschaft.
Der in Abb. 1.1 im Gesamtkontext eingebettete Reportingclient existiert für den eLettershop bereits in einer PHP-HTML-Variante. Dieser wird von Kunden zum

[2] "Kennzeichen für das Regeln ist der geschlossene Wirkungsablauf, bei dem die Regelgröße im Wirkungsweg des Regelkreises fortlaufend sich selbst beeinflusst." (vgl. DIN 19226 Teil 1)

Einen hinsichtlich der mangelnden Übersichtlichkeit kritisiert. Des Weiteren fehlen, nach Meinung der Anwender, essentielle grafische Vergleichsfunktionen. Diese sollen dem Auffinden von Kausalzusammenhängen dienen. Da die beiden wesentlichen Funktionen des Reportingclients die Visualisierung und Interaktion darstellen, bietet sich für seine technische Umsetzung eine Rich Internet Application[3] (RIA) an. Eine nähere Erläuterung ihres Charakters sowie der ihr zugrundeliegenden Technologien wird im nächsten Abschnitt gegeben. Dieser stellt zudem einen kurzen geschichtlichen Abriss der Entwicklung des World Wide Web von der Veröffentlichung der Sprache HTML bis hin zu den Rich Internet Applications dar.

1.2 Die Entwicklung des World Wide Web (WWW)

Die Bezeichnung World Wide Web wird häufig als Synonym für das Internet verwendet, obwohl es sich hierbei um lediglich einen von vielen Diensten (E-Mail, FTP etc.) des Internet handelt. Das World Wide Web in seiner heutigen Form, wurde im Jahre 1989 von dem britischen Informatiker Sir Timothy John Berners-Lee im Rahmen seiner Tätigkeit bei der Europäischen Organisation für Kernforschung entwickelt (vgl. [MS04]). Seit der Veröffentlichung im April 1993 erfuhr es eine rasante Entwicklung.
Um am WWW partizipieren zu können, wird neben einer Anbindung zum Internet ein Webbrowser[4] benötigt. Dieser stellt formatierte Texte und Grafiken dar, welche von Webservern geladen werden. Zudem ermöglicht er mit Hilfe von Hyperlinks – im Folgenden als “Links” bezeichnet – die einfache Navigation zwischen Webseiten. Zusätzliche Funktionalität wird durch die Verwendung von Browser-Plugins[5] erreicht. Diese erlauben z. B. die Wiedergabe von audiovisuellen Daten.
Die ersten Webseiten wurden vollständig mittels der von Tim Berners-Lee entwickelten Sprache HTML beschreiben und waren somit rein statisch. Die entscheidende Funktionalität dieser Sprache stellt die Möglichkeit dar, Verweise in Form von eingebetteten Links auf andere Dokumente (Webseiten) geben zu können. So konnte ein weltweites Netzwerk aus Webseiten entstehen. Wachsende Beliebtheit und die Aussicht auf eine zukünftige Kommerzialisierung förderten die Entwicklung verschiedener Technologien, welche den Webnutzern ansprechendere Webauftritte bieten sollten.
Die Abbildungen 1.2 und 1.3 zeigen die Webseiten von MTV[6] aus den Jahren 1997[7] und 2009[8]. Anhand dieses Beispiels ist der allgemeine Trend zu mehr Grafikeinsatz und gesteigerter Interaktivität erkennbar.

[3] engl. für reichhaltige Internet-Anwendung
[4] Programm um Webseiten betrachten und benutzen zu können. Bekannte Webbrowser sind z. B. Internet Explorer, Mozilla Firefox sowie Safari
[5] Browser-Plugins, teilweise auch “Add-ons” genannt, sind Computerprogramme und erweitern die Funktionalität eines Webbrowsers.
[6] MTV steht für “Music Television” und ist ein internationaler Musiksender im Fernsehprogramm.
[7] siehe `http://www.mtv.de`
[8] siehe `http://web.archive.org/web/19971015121450/http://www.mtv.com`

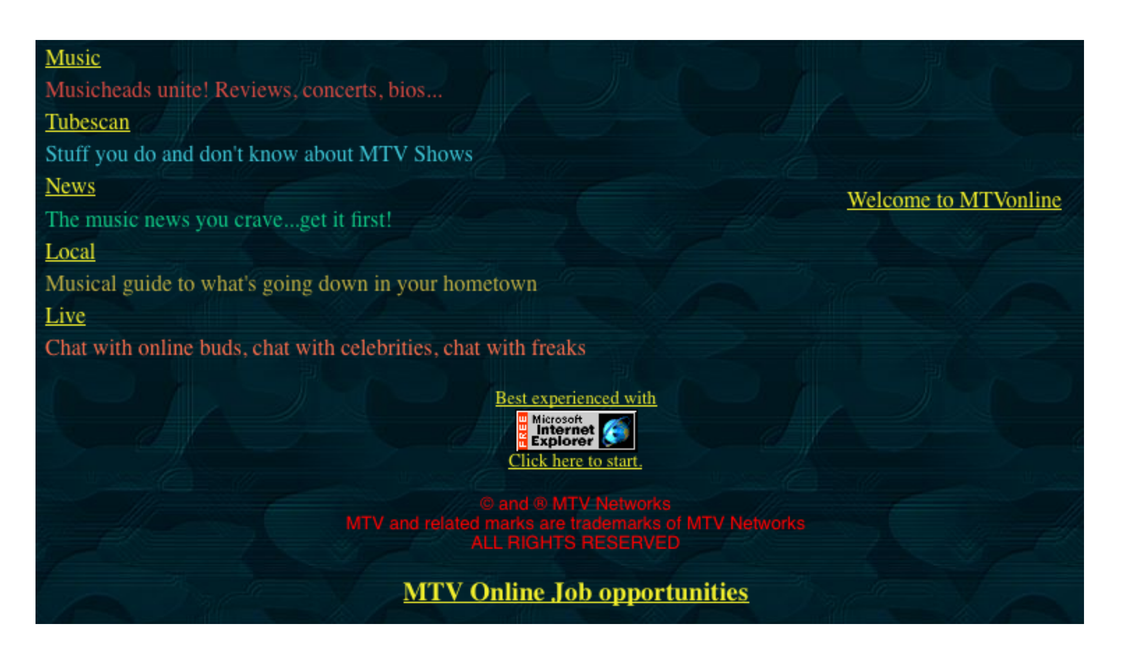


Abbildung 1.2: Internationaler MTV-Webauftritt 1997

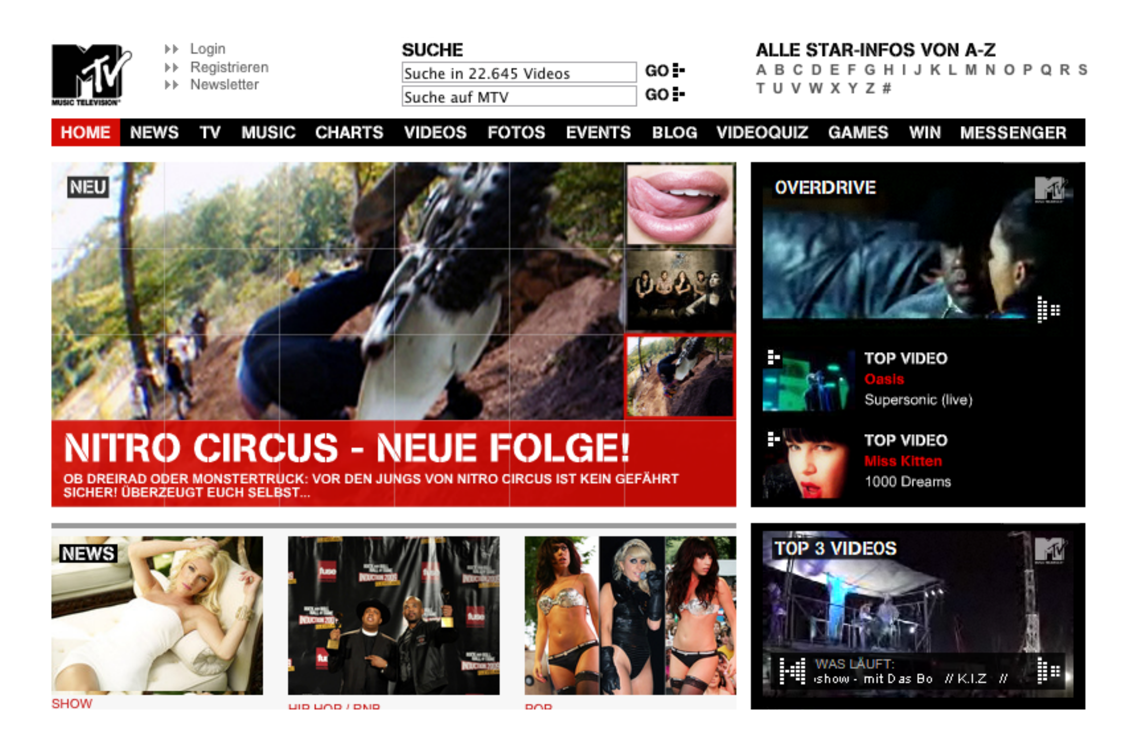


Abbildung 1.3: Deutscher MTV-Webauftritt 2009

Während HTML zur Abbildung statischer Inhalte bereits ausreicht, ist es für die Realisierung komplexerer Interaktionen ungeeignet. Daher entstanden schon recht früh erste Ansätze zum Aufbau dynamischer Webseiten (vgl. PHP, Abs. 2.1), welche auch die Einbindung von Datenbanken berücksichtigten.
Trotz der dadurch erreichten Interaktivität können konventionelle Webapplikationen noch nicht den Funktionsumfang von Desktop-Software bieten. Aus verschiedenen Gründen, u. A. dem der Plattformunabhängigkeit, zeichnet sich seit einigen Jahren der Trend zu desktop-ähnlichen Webapplikationen ab. Solche Webanwendungen fallen in die Rubrik Rich Internet Application.

> "In Rich-Internet Anwendungen (RIAs) vereinen sich die Funktionalität und der Komfort von klassischen Windows Desktop-Anwendungen mit dem breiten Einsatzspektrum und der preisgünstigen Bereitstellbarkeit von Webanwendungen."[9]

Eine frühe Entwicklung in diese Richtung initiierte SUN mit so genannten Java-Applets, welche sich jedoch nicht durchsetzen konnten.

> "We shouldn't forget that Java really started as an RIA technology back in the late 90s with applets."[10]

Ein Durchbruch gelang mit der Webtechnologie Ajax[11], die auf den bereits etablierten Technologien JavaScript und XML basiert. Mit Ajax ist es möglich, die Aktualisierung einer Webseite auf bestimmte Bereiche zu beschränken. Folgendes Zitat fasst die daraus hervorgehenden Vorteile zusammen.

> "Recently and rapidly, the combination of JavaScript and XML (Ajax) has improved the Web experience. Using Ajax, wholesale page reloads are unnecessary; feedback is more immediate; and browser-bound applications act more like desktop software."[12]

Zusätzliche Möglichkeiten der effizienten grafischen Darstellung wurden mit der Entwicklung der Flash-Technologie eingeführt. Anfangs wurden Flash-Dateien in Webseiten eingebettet und dienten der Wiedergabe bewegter Inhalte. Mittlerweile basieren einige Webangebote vollständig auf dieser Technik.
Die jüngste Entwicklung stellen Frameworks zur Realisierung von RIAs dar. Eines hiervon ist Flex, welches die performante Visualisierung der Flash-Technologie nutzt und diese um interaktive Bedienelemente ergänzt. Flex führt somit keine grundlegend neue Technologien ein, sondern kombiniert bereits vorhandene und vereinfacht ihre Anwendung.

[9] vgl. `http://www.ecomplexx.com/rich-internet-adobe-flex.aspx`, zugegriffen am 20. Februar 2009
[10] vgl. `http://ajax.sys-con.com/node/768626`, zugegriffen am 20. Februar 2009
[11] "Asynchronous JavaScript and XML" (Ajax) bezeichnet die asynchrone Datenübertragung zwischen Webbrowser und Webserver unter Verwendung von JavaScript und XML
[12] vgl. `http://www.linux-mag.com/id/3308`, zugegriffen am 3. März 2009

2 Grundlagen

In den folgenden Abschnitten sollen relevante Methoden, Konzepte, Technologien und Dienste für die vorliegende Arbeit beschrieben werden. Dazu gehören neben den technischen Grundlagen wie z. B. Programmiersprachen auch spezielle Informationen zum Bereich Reporting.

2.1 PHP: Hypertext Preprocessor

Um die Unzulänglichkeiten statischer Webseiten zu vermeiden, wurde bereits in einer sehr frühen Entwicklungsphase des WWW die Skriptsprache PHP entwickelt. Das rekursive Akronym steht für "PHP: Hypertext Preprocessor" und wird auf der offiziellen Webseite[1] wie folgt beschrieben:

> "[...] eine weitverbreitete Open Source Skriptsprache speziell für Webentwicklungen. PHP läßt sich in HTML einbinden. Die Syntax erinnert an C, Java und Perl und ist einfach zu erlernen. Das Hauptziel dieser Sprache ist es, Webentwicklern die Möglichkeit zu geben, schnell dynamisch generierte Webseiten zu erzeugen."[2]

PHP ist im Bereich der Webentwicklung stark verbreitet. Aktuell befindet sich die Sprache in der Version 5 und unterstützt mittlerweile die objektorientierte Programmierung. Es existieren zahlreiche vorgefertigte und frei benutzbare Module, die von Open Source Gemeinschaften erstellt wurden. Für viele Arten von Webseiten wie z. B. Blogs oder Online-Shops stehen komplett entwickelte Anwendungen kostenlos zum Download bereit.

2.2 Model-View-Controller (MVC)

Bei Model-View-Controller (MVC) handelt es sich um ein Programmierkonzept, welches sich für die Realisierung großer Softwareprojekte anbietet. MVC zeichnet sich im Wesentlichen durch die Trennung von Daten, Funktionen und Sichten sowie vorab definierte Schnittstellen aus. In Abb. 2.1 wird das Konzept skizziert, wobei

[1] siehe `http://www.php.net`
[2] vgl. `http://de.php.net/manual/de/preface.php`, zugegriffen am 20. Februar 2009

die durchgezogenen Linien direkte und die gestrichelten indirekte Assoziationen darstellen[3]. Durch diese Strukturierung lassen sich u. A. Inkonsistenzen, wie sie bei der

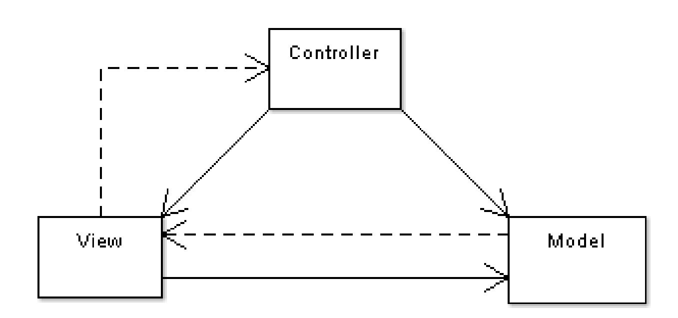


Abbildung 2.1: Model-View-Controller Konzept

gemeinschaftlichen Programmierung häufig auftreten, vermeiden. Dieses trägt zu einer effizienten Entwicklung, Wartung und Erweiterung der Software bei.

2.3 Zend Framework

Für umfangreiche Softwareprojekte bietet sich in der Regel ein Framework an. Es erleichtert die Programmierarbeit durch vorgefertigte, anpassbare Workflows. Meist werden Bibliotheken für Datenbankanbindungen oder Sitzungsmanagement bereitgestellt. Welches Framework im konkreten Fall genutzt wird, hängt von individuellen Präferenzen, der Programmiersprache sowie von der Art des Softwareprojektes ab. Web-Application-Frameworks sind speziell für dynamische Webseiten optimiert.
Das PHP-Backend des eLettershops basiert auf dem Zend Framework[4]. Dabei handelt es sich um eine Open Source Software, welche komplett in PHP 5 geschrieben ist. Neben diversen Paketen ist das bereits beschriebene MVC-Konzept implementiert (vgl. [ALB08]).

2.4 Adobe Flex

Bei Adobe Flex handelt es sich um ein Entwicklungsframework für die Erstellung von Rich Internet Applications. Mihai Corlan, Mitarbeiter von Adobe, beantwortete die Frage "What is Flex?" wie folgt:

> "Flex is just another way to create a Flash application"[5]

Die mit Flex erstellten Applikationen sind Flash-Dateien mit der Dateiendung ".swf", die mittels HTML eingebunden werden, um sie im Webbrowser darzustel-

[3] vgl. `http://de.wikipedia.org/wiki/Model_View_Controller`, zugegriffen am 4. Mai 2009
[4] siehe `http://framework.zend.com`
[5] vgl. `http://corlan.org/downloads/Flex_AIR_and_PHP.pdf`, zugegriffen am 23. März 2009

len. Das Adobe Flex SDK[6] bildet die Grundlage für Flex. Es stellt die Compiler[7], die Komponentenbibliothek sowie den Debugger[8] zur Verfügung. Mit der Veröffentlichung von Version 3 (Codename "Moxie") im Februar 2008, wurde der Quellcode von Adobe offengelegt[9]. Mit dem kostenlosen Open Source SDK und einem beliebigen Editor lassen sich bereits komplette Flex-Applikationen erstellen[10].

Das Flex SDK bietet dem Entwickler eine Klassenbibliothek, die häufig verwendete Standardkomponenten bereitstellt (vgl. [NA08]). Zusätzlich lassen sich eigene Komponenten ("Custom Components" genannt) entwerfen. Diese können als Quellcode sowie in kompilierter Form anderen Entwicklern zur Verfügung gestellt werden.
Adobe bietet neben der Entwicklung von Browserapplikationen noch die Option an, mit dem Flex SDK Desktopanwendungen zu realisieren. Hierfür wurde die plattformunabhängige Laufzeitumgebung "Adobe Integrated Runtime" (AIR) geschaffen. AIR-Dateien werden per Internetverbindung lokal auf dem Rechner installiert und können wie herkömmliche Desktopanwendungen offline ausgeführt werden.
Aktuell ist Flex stabil in Version 3.1 (Stand April 2009) erhältlich. Der Nachfolger, Flex 4 (Codename "Gumbo"), besitzt den Beta-Status.

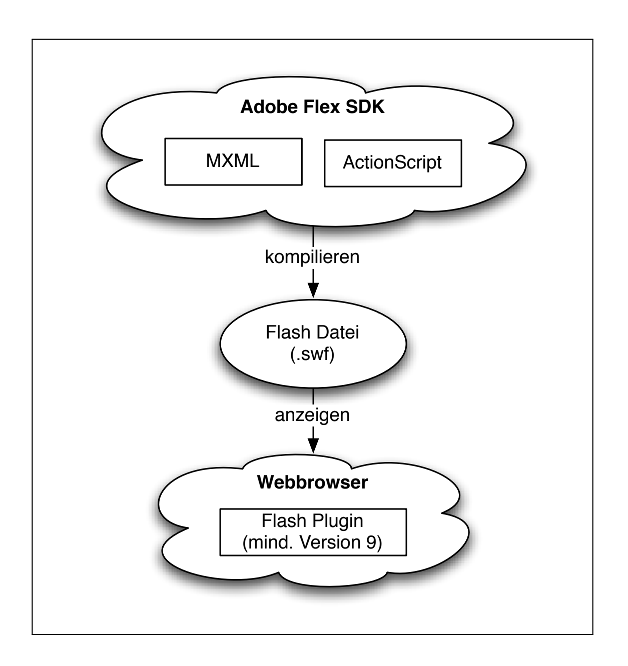


Abbildung 2.2: Flex Architektur

[6] Unter einem "Software Developer Kit" (SDK) versteht man im Allgemeinen eine Kollektion von Hilfsprogrammen, welche der Software-Entwicklung dienen.
[7] engl. für Kompilierer; Ein Computerprogramm, das den Quellcode in das gewünschte semantisch äquivalente Zielprogramm umwandelt.
[8] Ein Debugger ist ein Werkzeug mit der Aufgabe, Fehler in einem Programm zu finden und zu beheben.
[9] siehe `http://opensource.adobe.com`
[10] siehe `http://labs.adobe.com/technologies/flex/sdk`

Abbildung 2.2 veranschaulicht die Flex-Architektur. Flex kombiniert MXML und ActionScript. Grundsätzlich ist es mit beiden Sprachen möglich, Komponenten (z. B. Bedien- und Anzeigeelemente) zu erzeugen. Eine nähere Erläuterung erfolgt in den nächsten beiden Abschnitten.

2.4.1 MXML

MXML ist eine beschreibende, auf XML[11] basierende Programmiersprache. Ihr Code wird zur Kompilierungszeit vom Flex-Compiler in ActionScript-Code übersetzt (vgl. [Wid08]). Zwar ist MXML für die Objekterstellung in Flex nicht zwingend notwendig, systematisiert jedoch den Entwicklungsprozess. Indem versucht wird, den Quellcode weitgehend in funktionelle und beschreibende Teile zu gliedern, wird die Übersichtlichkeit positiv beeinflusst.
Im folgenden Beispiel wird ein Button mit der Aufschrift “Klick mich!” erstellt.

```
1 <mx:Button id="button1" //Komponente
2   x="10" y="10" //Positionierung
3   label="Klick mich!" //Beschriftung
4   click="handleClick()"/> //Funktion
```

Quellcode 2.1: Beispiel MXML-Komponente

Für den funktionellen Part (`handleClick()`) ist die befehlsorientierte Programmiersprache ActionScript verantwortlich, welche im nächsten Abschnitt beschrieben wird.

2.4.2 ActionScript

Adobe Flex benutzt die befehlsorientierte Programmiersprache ActionScript, welche sich derzeit in Version 3 befindet. Sie stellt ein Derivat der Sprache JavaScript dar und ähnelt ihr somit in der Syntax. In Flex und Flash wird sie verwendet um das Verhalten der Applikation festzulegen. Im Gegensatz zu MXML, erlaubt ActionScript das Hinzufügen und Modifizieren zur Laufzeit. Der Großteil der Verbindungstechnologie zum PHP-Backend des Reportingclients wird mit Hilfe von ActionScript realisiert.
Folgendes Beispiel legt die Aktion fest, die durch Klicken des im Code-Beispiel 2.1 erstellten Buttons ausgelöst wird. In diesem Fall wird lediglich ein Fenster, resp. Alert-Box mit dem Text “Geklickt!” ausgegeben.

```
1 private function handleClick():void {
2   Alert.show("Geklickt!"); //zeigt eine Alert-Box
3 }
```

Quellcode 2.2: Beispiel ActionScript-Funktion

[11] XML steht für “Extensible Markup Language” und bezeichnet ein flexibles Textformat.

2.4.3 Flex-Builder

Adobe bietet neben dem kostenlosen Eclipse[12]-Plugin noch die kostenpflichtige Applikation “Flex-Builder” an. Mit dieser integrierten Entwicklungsumgebung soll die Aggregation von Flex-Komponenten durch “Drag and Drop”[13]-Funktionalität vereinfacht werden. Zusätzlich zu der normalen Quellcode-Ansicht existiert die Design-Ansicht, die einem WYSIWYG[14]-Editor gleicht. Neben einem vereinfachten Zugang für Einsteiger bietet die Option der grafischen Programmierung (z. B. Drag and Drop) die automatische Codegenerierung bei Standardaufgaben. Dieses Vorgehen ist in der Regel zeiteffizienter und weniger fehleranfällig als die manuelle Programmierung. Intuitiver gestaltet sich die Arbeit zudem durch die Verwendung von Parametermasken. Diese zeigen dem Anwender direkt die zur Verfügung stehenden Funktionen.
Beim Erstellen eines neuen Flex-Projekts wird der Entwickler durch einen Wizard unterstützt. Dieser fragt den Anwender, welcher Applikationstyp erstellt werden soll. Neben einer Webanwendung, die im Flash-Player läuft, wird die Option angeboten, eine Desktopanwendung zu erstellen, welche mit Adobe AIR ausgeführt werden kann.

2.4.4 Flex-Charting

Als Erweiterung zum Flex SDK erlauben es die kommerziellen Flex-Charting-Komponenten, Diagramme effizient zu implementieren. Beispielsweise sind vordefinierte Kreis- und Liniendiagramme verfügbar, welche für den zu realisierenden Reportingclient genutzt werden. Das folgende Code-Beispiel 2.3 zeigt anschaulich, wie effizient die Erstellung eines Kreisdiagramms mit Hilfe von MXML erfolgen kann.

```
1  <mx:PieChart id="pieChart1" //Komponente
2        dataProvider="{pieDataProvider}" //Datenbereitstellung
3    <mx:series>
4      <mx:PieSeries id="pieSeries1" //Datensatz
5          field="@data"> //Datenfeld
6      </mx:PieSeries>
7    </mx:series>
8  </mx:PieChart>
```

Quellcode 2.3: Beispiel Flex Kreisdiagramm

[12] Eclipse ist ein Open Source Editor mit Unterstützung für viele Programmiersprachen (siehe http://www.eclipse.org).
[13] engl. für ”Ziehen und Fallenlassen“; bezeichnet eine Methode um auf einer Benutzeroberfläche Elemente durch Bewegung zu bedienen.
[14] “What you see is what you get” (WYSIWYG) bezeichnet eine Erstellungsvariante, bei der das Endergebnis schon direkt beim Erstellen sichtbar ist.

2.5 E-Mail

Da sich das Reporting intensiv mit der Datenauswertung von E-Mail-Nachrichten befasst, soll der Internetdienst E-Mail in diesem Abschnitt näher betrachtet werden. Einen Überblick zur grundsätzlichen technische Umsetzung dieses Dienstes gibt die abstrahierte Darstellung in Abb. 2.3.

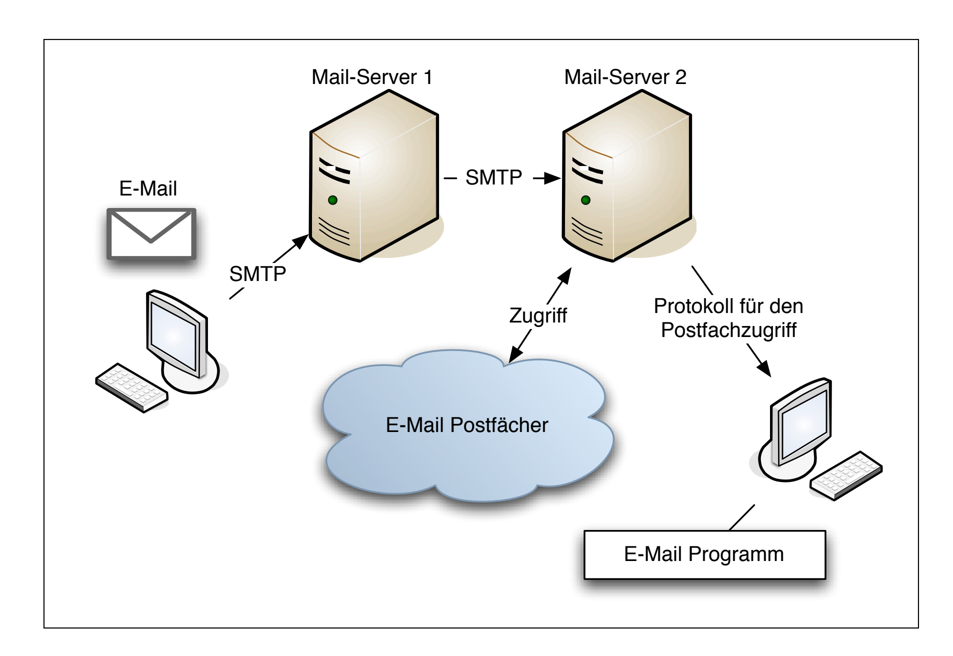


Abbildung 2.3: Die Übermittlung einer E-Mail

Neben dem Protokoll für das Sendeverfahren (SMTP), werden noch die beiden internen Bestandteile einer E-Mail (Header und Body) erläutert.

2.5.1 Simple Mail Transfer Protocol (SMTP)

Der E-Mail-Versand ist durch das "Simple Mail Transfer Protocol" (SMTP) definiert. Dabei erfolgt der Datenaustausch nicht direkt zwischen Sender und Empfänger, sondern über spezielle Mail-Server, welche die versendeten Daten entgegennehmen. Der Abruf erfolgt mit Hilfe eines E-Mail-Programms unter der Verwendung eines Übertragungsprotokolls. Hierbei stellen POP[15] und IMAP[16] zwei standarisierte Protokolle für den E-Mail-Empfang dar.[17]

[15]Das "Post Office Protocol" (POP) ist ein Übertragungsprotokoll für den E-Mail-Empfang.
[16]Das "Internet Message Access Protocol" (IMAP) ist ein Anwendungsprotokoll für den Zugriff und die Verwaltung von E-Mails auf Mailservern.
[17]vgl. `http://www.webopedia.com/TERM/S/SMTP.html`, zugegriffen am 3. März 2009

2.5.2 Kopfzeilen (Header)

Der E-Mail-Header besteht aus mehreren für den Empfänger relevanten Daten. Er kann mit den Informationen eines Briefumschlages verglichen werden. In der Regel enthält der Header neben Absender- und Empfängeradresse zusätzliche Informationen wie z. B. den Zeichensatz des Bodys (siehe Abs. 2.5.3), die durchlaufenen E-Mail-Server oder die Antwortadresse. Neben einigen wenigen Pflichtfeldern sind die meisten Angaben im Header optional[18]. Für das Reporting wird ein spezieller Parameter, die Mailingidentifikationsnummer (MailingId), gesetzt. Dieser erlaubt das Zuordnen jeder einzelnen E-Mail zu einem konkreten Mailing (siehe Abs. 2.6.1).

2.5.3 Inhalt (Body)

Im E-Mail-Body ist der eigentliche Inhalt der E-Mail gespeichert. Es existieren drei verschiedene Formate, in denen eine E-Mail verschickt werden kann.

- **Text**
  Beim Text-Format handelt es sich um Inhalt in unformatiertem Text. Bilder können lediglich als Anhang versendet werden. Die Formatierungen des Textes werden durch das E-Mail-Programm des Empfängers bestimmt.

- **HTML**
  E-Mails im HTML-Format erlauben die Übermittlung von formatiertem Text und eingebundenen Grafiken. Bei diesem Format ist zu beachten, dass nicht jedes E-Mail-Programm die Anzeige von HTML unterstützt.

- **Multipart**
  Eine Mischform stellt "Multipart" dar, bei dem die E-Mail in beiden der oben beschriebenen Formate übermittelt wird. In welcher Form die Darstellung auf der Empfängerseite erfolgt, hängt von der Konfiguration des verwendeten E-Mail-Programms ab. Ist eine Darstellung im HTML-Format nicht möglich, so wird im Allgemeinen das Text-Format angezeigt. E-Mails die nur im HTML-Format vorliegen, lassen sich in diesem Fall nicht anzeigen.

2.6 Reporting

Bevor auf den Versand von E-Mails und deren Auswertung eingegangen wird, ist zunächst der Begriff "Reporting" im Rahmen dieses Projektes zu definieren. Er bezeichnet die Erhebung, Speicherung, Verarbeitung, Analyse und Darstellung von Daten bezüglich des E-Mail-Versands. Mit Reportingfunktionen sind Darstellungen von gespeicherten und evtl. verarbeiteten Reportingdaten gemeint.

[18] vgl. `http://tools.ietf.org/html/rfc5322`, zugegriffen am 3. März 2009

2.6.1 Mailing

Der Begriff "Mailing" bezeichnet die komplette Abwicklung des E-Mail-Versands an eine definierte Empfängerliste. Es setzt sich neben dem E-Mail Header (z. B. Absenderangaben oder personalisierter Betreff) hauptsächlich aus folgenden beiden Komponenten zusammen:

- **Empfängerliste**
  Dem Kunden wird ermöglicht, eine Empfängerliste in Tabellenform zu importieren, die neben den E-Mail-Adressen noch Namen und weitere personenspezifische Daten enthalten kann. Diese lassen sich dazu verwenden, den Inhalt zu personalisieren und die E-Mail dadurch individueller zu gestalten.

- **Personalisierter Inhalt**
  Die genannte Personalisierung wird durch Platzhalter realisiert (z. B. "Hallo [VORNAME],"). Diese werden unmittelbar vor dem Versand mit Hilfe einer Routine gegen die Daten aus der Empfängerliste ersetzt. Diese Funktionalität stellt die notwendige Basis für das in der Motivation beschriebene Optimierungsverfahren dar. Ziel ist es, dass versendete E-Mails möglichst hohe und positive Resonanz erzeugen. Dies hängt u. A. von der Gestaltung der E-Mail ab.

Die Basis für die Reportingfunktionen bildet das Tracking, welches im folgenden Abschnitt erläutert wird.

2.6.2 Tracking

Um das Verhalten der E-Mail-Empfänger zu analysieren, muss deren Handlung zunächst protokolliert werden. Die notwendige Methode dafür nennt sich "Tracking"[19]. Unter diesem Begriff versteht man das Beobachten und Speichern relevanter Aktionen der Empfänger. Abb. 2.4 veranschaulicht den prinzipiellen Tracking-Ablauf. Der Prozess beginnt mit einem vom E-Mail-Empfänger ausgelösten Ereignis. Dieses wiederum führt zu einer Anfrage an einen Webserver, welche jedoch zunächst über einen Tracking-Server läuft. Der Tracking-Server observiert die Empfängeraktionen und legt diese in der Reporting-Datenbank ab. Es existieren verschiedene Tracking-Informationen, die das Reporting wiedergeben soll. Um eine klare Differenzierung der einzelnen Empfängeraktionen zu erreichen, werden in der folgenden Auflistung die möglichen Aktionen voneinander abgegrenzt.

- **E-Mail übermittelt**
  Der Empfang einer E-Mail lässt sich nicht immer verifizieren. Andererseits informiert ein auftretender Bounce (siehe Abs. 2.6.3) über eine nicht zugestellte E-Mail. Eine eindeutige Empfangsbestätigung stellt das Öffnen der E-Mail bzw. Klicken auf einen der in der E-Mail enthaltenen Link dar.

[19] engl. für Spurbildung

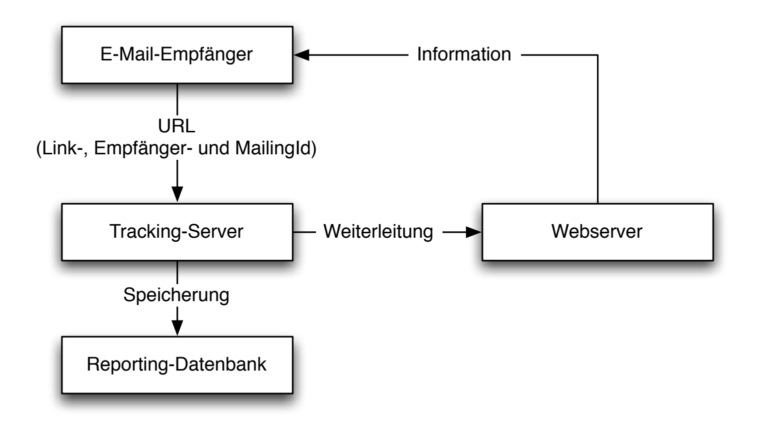


Abbildung 2.4: Tracking-Ablauf

- **E-Mail geöffnet**
  Ein mögliches Verfahren, das dem Sender den erfolgreichen E-Mail-Versand bestätigen kann, ist die in E-Mail-Programmen wählbare Option "Lesebestätigung anfordern". Diese Lesebestätigung kann jedoch vom Empfänger unterdrückt werden, wodurch diese Option an Aussagekraft verliert.
  Eine weitere Möglichkeit, die Öffnung zu verifizieren, ergibt sich über das Einbinden einer Grafik, die über den Tracking-Server geladen wird. Es genügt ein transparenter Pixel, der keinen nennenswerten Traffic verursacht. Diese Grafik wird über eine URL[20] eingebunden, welche einen personalisierten Schlüssel enthält. Er setzt sich aus der ID des Empfängers und des Mailings sowie der Zieladresse zusammen. Hierdurch kann das Öffnen einer E-Mail eindeutig einem bestimmten Empfänger zugeordnet werden. Diese Parameter werden verschlüsselt übermittelt, um das Manipulationsrisiko der Reportingdaten zu minimieren.
  Wie auch die Lesebestätigung, lässt sich das automatische Laden von Bildern in E-Mail-Nachrichten unterdrücken, so dass eine Öffnung zu keiner Rückmeldung führt. Beide genannten Methoden sind unzureichend, um verlässliche Aussagen treffen zu können.
  Durch Klicken auf einen beliebigen Link in der E-Mail bestätigt der Empfänger sowohl den Empfang als auch die Öffnung der E-Mail. Dieses Verfahren wird im nächsten Punkt beschrieben.

- **Link in der E-Mail angeklickt**
  Die Ermittlung der Klicks auf Links entspricht prinzipiell dem bereits beschriebenen Öffnungs-Tracking mittels einer eingebetteten Grafik. Jeder Link besitzt eine Identitätsnummer und wird über den Tracking-Server geroutet. Den Empfängern wird ebenfalls eine eindeutige Id zugeordnet, so dass konkret

[20]URL ist die Abkürzung für "Uniform Resource Locator", umgangssprachlich auch "Internetadresse" genannt

bestimmt werden kann, wer zu welchem Zeitpunkt auf welchen Link geklickt hat. Zusätzlich wird in diesem Fall der Öffnungsstatus aktualisiert, falls das Tracking-Bild nicht geladen wurde. Diese Informationen werden vom Tracking-Server in die Reporting-Datenbank geschrieben, bevor der Anwender zu der gewünschten Internet-Adresse weitergeleitet wird.

Unter gewissen Umständen erreichen E-Mails nicht den gewünschten Empfänger. Hierfür kann es verschiedene temporäre oder permanente Gründe geben, die im folgenden Abschnitt näher erörtert werden.

2.6.3 Bounces

E-Mails erreichen nicht immer den gewünschten Empfänger, da z. B. die Quota[21] des Postfachs erreicht ist oder der E-Mail-Server die E-Mail nicht entgegennimmt. Nicht vollständig übermittelte E-Mails fallen in die Kategorie Bounce[22]-Messages (im Folgenden nur noch Bounces genannt). Man unterscheidet zwei verschiedene Formen; Soft und Hard Bounces, die allerdings unterschiedlich von E-Mail-Servern klassifiziert werden. Eine allgemeingültige Definition dieser Bounce-Arten existiert nicht. Die Entwickler des eLettershop spezifizieren diese wie folgt:

- **Soft Bounce**
  Bei Soft Bounces handelt es sich um temporäre Zustellprobleme. So kann es vorkommen, dass eine E-Mail den E-Mail-Server des Empfängers erreicht, die Empfängeradresse somit gültig ist, diese aber dennoch vom Server zurückgewiesen wird. Gründe hierfür können u. A. eine temporäre Serverüberlastung oder eine Überschreitung der erlaubten E-Mail-Größe sein.

- **Hard Bounce**
  Ein Hard Bounce wird verursacht, wenn ein permanentes Problem mit der Empfänger-Adresse besteht. Ein häufig auftretender Grund dafür stellen ungültige E-Mail-Adressen dar. Falls die Domäne nicht existiert oder Schreibfehler in der Adresse auftreten, wird die E-Mail in der Regel direkt vom E-Mail-Server zurückgewiesen.

Im Reporting werden die fehlgeschlagenen Versendungen lediglich in Soft und Hard Bounces kategorisiert, da eine detaillierte Auflistung der einzelnen Bounce-Gründe im Reportingclient die Übersichtlichkeit unterminiert.

2.6.4 Status

Eine wesentliche Eigenschaft eines Mailings ist dessen Status. Er gibt an, in welchem Stadium es sich befindet. Nachfolgend sind alle für das Reporting relevanten Zustände chronologisch gelistet:

[21] Quota bezeichnet in diesem Fall die Begrenzung des Speicherplatzes eines E-Mail-Postfachs.
[22] engl. für abprallen

- **DISCOVERED**
  Ein neues Mailing wurde in das System eingebracht. Dies kann über mehrere Schnittstellen geschehen. Einige Kunden geben die relevanten Daten über die Webschnittstelle ein, andere bevorzugen die Übermittlung per FTP[23].
- **IMPORT_RUNNING**
  Hier wird der Inhalt der E-Mail eingelesen.
- **IMPORT_FINISHED**
  Die Übermittlung von E-Mail-Body und -Header an die Datenbank ist in diesem Status abgeschlossen.
- **IMPORT_RECIPIENT_RUNNING**
  Die Empfängerliste wird standardmäßig per ".csv"-Datei übertragen. Sie enthält neben E-Mail Adresse meist empfängerspezifische Daten, die mit in die personalisierte E-Mail eingebunden werden.
- **IMPORT_RECIPIENT_FINISHED**
  Dieser Status gibt an, dass die Empfängerliste eingelesen wurde und das Mailing versendet werden kann.
- **SHIPMENT_PLANNED**
  Der Versandzeitpunkt für das Mailing wurde festgelegt. Bis zum Versand können noch Änderungen an Inhalt und Empfängerliste vorgenommen werden.
- **SHIPMENT_RUNNING**
  Der E-Mail-Versand ist aktiv.
- **SHIPMENT_FINISHED**
  Der Versandprozess wurde abgeschlossen. Das Mailing ist abgearbeitet.

2.7 Data-Warehouse

Data-Warehouse bezeichnet eine spezielle Art von Datenbank, welche ihre Daten aus mehreren Quellen bezieht und der weiteren Datenanalyse dient (vgl. [BG08]). Ziel dieser Analyse ist das Auffinden von Kausalzusammenhängen. Solche Korrelationen sind speziell für das Reporting von Bedeutung, da sie Entwicklungen herausstellen können und Grundlage für Marketingstrategien bilden. Diese Behauptung lässt sich an einem projektnahen Beispiel verdeutlichen, welches auf der Datenanalyse mittels eines Data-Warehouse basiert. Laut dieser werden in E-Mails eingebettete Grafiken von E-Mail-Programmen zunehmend unterdrückt. Da die Anbieter von E-Mail Diensten

> "[...] kontinuierlich nach neuen Wegen zur Abwehr von Spam, Viren und Spyware suchen, hat sich eine anfänglich kaum verbreitete Funktion –

[23] Das "File Transfer Protocol" (FTP) ist ein Protokoll für die Dateiübertragung im Internet.

> die standardmäßige Unterdrückung der Anzeige von Grafiken – in ein allgegenwärtiges Phänomen in der Auslieferungslandschaft für eMails verwandelt. Vermarkter, die ihre Kampagnen optimieren möchten, müssen über diese Praxis und deren möglichen Einfluss auf ihre seriöse eMail-Kommunikation Bescheid wissen."[24]

Allgemein gilt, dass sich mit zunehmendem Umfang der zugrundeliegenden Datensätze die Belastbarkeit der Analyseergebnisse verbessert.

[24]vgl. `http://www.ecin.de/marketing/bildunterdrueckung`, zugegriffen am 13. April 2009

3 Untersuchung

In den folgenden Abschnitten werden die Eigenschaften einer Flex-Lösung mit ihren Vor- und Nachteilen erörtert und Alternativen aufgezeigt. Durch eine gezielte Recherche nach bekannten Problemen können diese bereits in der Konzeptionsphase berücksichtigt und spätere Fehlentwicklungen vermieden werden. Dabei liegt das Hauptaugenmerk auf den möglichen Schnittstellen zwischen Flex und PHP. Neben der generellen Funktionsweise werden unterschiedliche Implementierungshilfen und Frameworks verglichen.

3.1 Eignung von Flex

Nachdem in Abschnitt 2.4 bereits grundlegende Informationen zu Flex vermittelt wurden, sollen nun wesentliche Vor- und Nachteile von Flex-Applikationen diskutiert werden. Zu berücksichtigen ist, dass es sich bei dem Ausgabeformat von Flex um eine Flash-Datei handelt, weshalb auch dieser Bereich zu betrachten ist.

3.1.1 Vorteile

Die Verwendung von Flex wurde durch den Auftraggeber vorgegeben, der seine Entscheidung wie folgt begründet:

- **Optik**
  Flash-Applikationen zeichnen sich besonders dadurch aus, dass vektorbasierte[1] Grafiken verwendet werden können. Dadurch kann eine sehr effiziente Übertragung und Darstellung von Bildinhalten und Animationen realisiert werden. Zudem sind stufenlose Transformationen (z. B. Skalierung, Rotation etc.) ohne Qualitätsverluste möglich.

- **Performance**
  Ein wesentlicher Aspekt von Benutzerfreundlichkeit sind kurze Antwortzeiten. Diese ergeben sich dadurch, dass die Flex-Applikation clientseitig verarbeitet resp. gerendert[2] wird. Somit reduzieren sich sowohl die Serverlasten bei Interaktionen innerhalb der Applikation als auch die notwendige Kommunikation. Im Standardfall werden zunächst alle benötigten Informationen auf den Client

[1] Vektorgrafiken werden im Gegensatz zu Rastergrafiken nicht mit Pixeln dargestellt, sondern mit Hilfe von Linien, Polygonen, Kurven und Flächen beschrieben.

[2] Rendern bezeichnet in diesem Fall das Berechnen und Darstellen der anzuzeigenden Objekte.

übertragen, wodurch im Anschluss daran keine weitere Serverkommunikation mehr stattfindet muss. Bei PHP-Webseiten ohne Ajax-Implementierung oder ähnlichen Ansätzen wird der Quellcode bei Interaktionen stets komplett neu geladen.

- **Sicherheit**
  Ein spezielles Problem des Internets ist das Phänomen des so genannten Phishing[3]. Dies geschieht häufig durch Nachahmung einer Seite und ähnlich aussehender Internetadresse. HTML-Seiten können mittels stets vorhandener Quelltextanzeige dupliziert werden. Der Nutzer gibt seine geheimen Daten in das vermeintlich authentische Frontend ein. Die Entwickler der Phishingseite gelangen so an die sensiblen Anwenderdaten, um diese anschließend zu missbrauchen. Bei Flash-Dateien ist der Identitätsdiebstahl durch das Nachbilden einer Webseite mit wesentlich größerem Aufwand verbunden, da die Option der Quelltextanzeige deaktivierbar ist. Zudem können Multimedia-Dateien in die Flash-Datei eingebunden werden. Für Contentanbieter bietet Flash darüber hinaus die Möglichkeit, ihre Inhalte vor Vervielfältigung effektiver zu schützen.

- **Kompatibilität**
  Viele Webentwickler kämpfen mit unterschiedlichen Darstellungsstandards in Webbrowsern. Es werden Stylesheets[4] verwendet, die jedoch teilweise browserabhängig dargestellt werden. Dieses Verhalten der Browser begründet sich in unterschiedlicher Interpretation von Darstellungsparametern oder fehlender Unterstützung einzelner Formatierungsoptionen (z. B. ist die Eigenschaft "border-spacing" für den Internet Explorer unbekannt[5]). Da Flash-Applikationen mit Hilfe eines Plugins ausgeführt werden, ergeben sich die genannten Darstellungsprobleme in diesem Fall nicht. Laut Adobe war der Flash-Player 9 im Dezember 2008 in ca. 98%[6] der in Europa installierten Webbrowser eingebunden.

- **Lizensierung**
  Die Version 3 des Flex SDKs ist Open Source und kostenlos. Durch die jahrelange Präsenz von Flex konnten sich bereits Entwicklergemeinschaften bilden, die weitere quelloffene Komponenten für Flex anbieten. Mit Hilfe des bestehenden Portfolios wird die Realisierung von Rich Internet Applications beschleunigt.

---

[3] Phishing ist eine Methode die ihren Namen aus "Password" und "Fishing" bekommen hat und auf das Ausspionieren von Passwörtern zielt.
[4] Bei Stylesheets handelt es sich um Dateien oder Text mit Formatierungsangaben.
[5] siehe `http://www.css4you.de/browsercomp.html/standardbrowser`
[6] vgl. `http://www.adobe.com/products/player_census/flashplayer/version_penetration.html`, zugegriffen am 21. April 2009

3.1.2 Nachteile

Nach Aussage von Jakob Nielsen im Oktober 2000

> "Flash: 99% bad"[7]

sind für ihn 99% aller Flash-Applikationen in Bezug auf Usability "schlecht". Nielsen begründet und relativiert seine Behauptung wie folgt:

> "None of these usability problems are inherent in Flash. You can design usable multimedia objects that comply with the guidelines and are easy to use. The problem is simply that current Flash design tends to encourage abuse."

Das Problem für Nielsen liegt in den umfangreichen grafischen Möglichkeiten einer Flash-Applikation und dem Reiz, diese übermäßig zu verwenden. Effekthascherei und ungewöhnliche grafische Benutzerobjekte mindern für ihn die Usability. Die Probleme der Benutzertauglichkeit sind jedoch nicht in Flash verwurzelt. Grundsätzlich lassen sich Flash-Applikationen erstellen, die die Richtlinien der Benutzerfreundlichkeit erfüllen.
Um Nielsens Kritik zu begegnen, soll bei der Konzeption besonders auf intuitive Benutzung Wert gelegt werden. Der Zweck der zu realisierenden Applikation, nämlich die anschauliche Visualisierung von Reporting-Daten, entkräftet Nielsens Argumentation. In diesem speziellen Fall macht man sich eben diese grafischen Optionen zu Nutze. Die detaillierte Ausarbeitung bezüglich des Designs wird im Kapitel 4 vorgestellt. Im Folgenden werden jedoch zunächst bekannte Flash- und Flex-Probleme aufgelistet und in Bezug zu diesem Projekt gestellt.

- **Flash-Plugin erforderlich**
  Obwohl die Kompatibilität aufgrund der Plattformunabhängigkeit als positiv dargestellt wird, ist es dennoch nachteilig, wenn der verwendete Browser zusätzliche Voraussetzungen zu erfüllen hat. Trotz der großen Verbreitung des Flash-Plugins sind nicht alle Nutzer erreichbar. Für die Browser einiger internetfähiger Mobil-Telefone sind bislang noch keine Flash-Plugins verfügbar.

- **Der Flex-Builder ist kostenpflichtig**
  Wie in Abschnitt 2.4.3 bereits erwähnt, lässt sich der WYSIWYG-Editor Flex-Builder nur mit Lizenz nutzen. Für Studenten gibt es die Möglichkeit, durch Immatrikulationsnachweis eine kostenlose Lizenz zu erhalten.

3.2 Alternativen zu Flex

Neben Adobe existieren zwei weitere große Anbieter auf dem Markt für RIA-Entwicklungsframeworks, namentlich Microsoft mit dem Produkt "Silverlight"[8] und SUN[9] mit "JavaFX". Im Folgenden werden diese beiden gängigsten Alternativen

[7] vgl. `http://www.useit.com/alertbox/20001029.html`, zugegriffen am 20. Februar 2009
[8] siehe `http://silverlight.net`
[9] siehe `http://javafx.net`

zu Flex kurz dargestellt. Die Code-Beispiele sollen einen Eindruck von Syntax und Semantik der jeweiligen Sprache vermitteln.

3.2.1 Silverlight

Microsofts Silverlight bezeichnet sowohl das Entwicklungsframework als auch das Browser-Plugin, das für die gängigsten Webbrowser verfügbar ist. Das Framework ermöglicht die plattformübergreifende Programmierung von .NET-Anwendungen[10]. Hierbei ist der Webentwickler nicht auf eine konkrete Sprache festgelegt. Silverlight erlaubt u. A. den Einsatz von JavaScript und Python[11].

Die Oberfläche, d. h. Anzeige- und Bedienelemente, wird mit Hilfe der Sprache XAML (eXtensible Application Markup Language), also letztlich in einem XML-Format beschrieben. Durch den grundsätzlichen Ansatz, Darstellung und Funktion zu trennen, wird bei Silverlight die Bindung an eine bestimmte Programmiersprache vermieden.

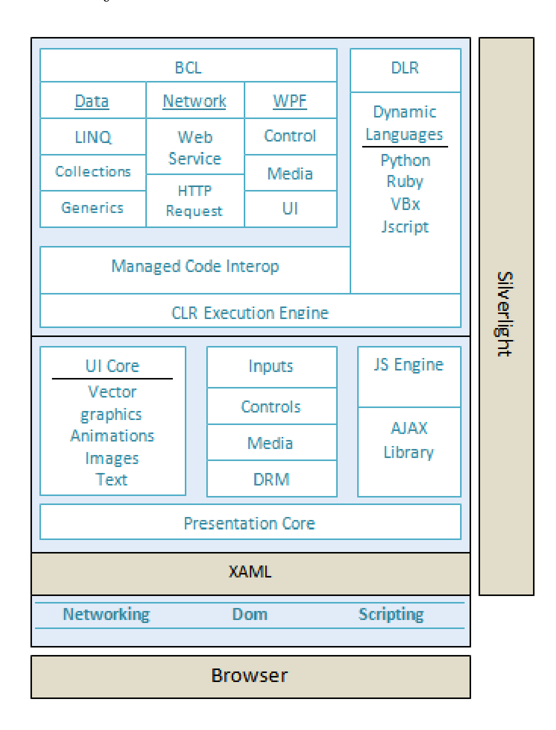


Abbildung 3.1: Silverlight Architektur[12]

Neben dem kostenlosen Framework und Plugin bietet Microsoft für die Erstellung einzelner Elemente in XAML den kostenpflichtigen Editor "Expression Blend 2"[13] an.

Aktuell befindet sich Silverlight in der Version 2.0. Seit März 2009 ist die Version 3.0 im Beta-Status erhältlich[14].

Als ein Produkt vom Softwarehersteller Microsoft unterstützt Silverlight nativ das Design einer Windows-konformen Benutzerschnittstelle. Dadurch vereinfacht sich die Entwicklung von RIAs im Microsoft Windows Look and Feel[15].

Folgendes Code-Beispiel zeigt die Erstellung eines Buttons in der Sprache XAML.

```
1  <UIControls:Button //Komponente
2    Canvas.Left="10" Canvas.Top="10" //Positionierung
3    Text="Klick mich!" //Beschriftung
4    Click="Funktionsname"/> //Funktion
```

Quellcode 3.1: Beispiel Silverlight-Button

Die Silverlight Architektur wird in Abb. 3.1 dargestellt.

[10] siehe http://www.microsoft.com/NET
[11] siehe http://www.python.org
[12] vgl. http://de.wikipedia.org/wiki/Microsoft_Silverlight, zugegriffen am 21. April 2009
[13] siehe http://www.microsoft.com/germany/expression/products/Overview.aspx?key=blend
[14] siehe http://www.microsoft.com/germany/expression/silverlight
[15] Die Bezeichnung "Look and Feel" steht für Designkonformität.

3.2.2 JavaFX

SUN als Entwickler der weit verbreiteten Programmiersprache Java veröffentlichte im Dezember letzten Jahres JavaFX in Version 1.0[16]. Wie der Name bereits vermuten lässt, basiert dieses Entwicklungsframework für RIAs auf der objektorientierten Open Source Sprache Java. In einem Artikel der PC Welt heißt es:

> "'Sun kommt etwas spät – aber nicht zu spät', meint IDC-Analyst Rüdiger Spies. Er räumt JavaFX Chancen im Kampf um den Web-2.0-Markt aufgrund seiner Offenheit und des Trend zu Software-as-a-Service (SaaS)[17] ein."[18]

SUNs Aktivitäten auf dem Markt für RIA-Entwicklungsframeworks sind jünger als jene der Konkurrenten Adobe und Microsoft.

> "'Der zu erwartende Boom bei SaaS macht die Notwenigkeit von interaktiven Web-Anwendungen nur größer', betont allerdings Spies. Suns Neuerung komme da gerade rechtzeitig. 'Damit können die vorhandenen Java-Anwendungen in SaaS-Umgebungen aufgepeppt werden', sieht der Analyst eine Chance für JavaFX."

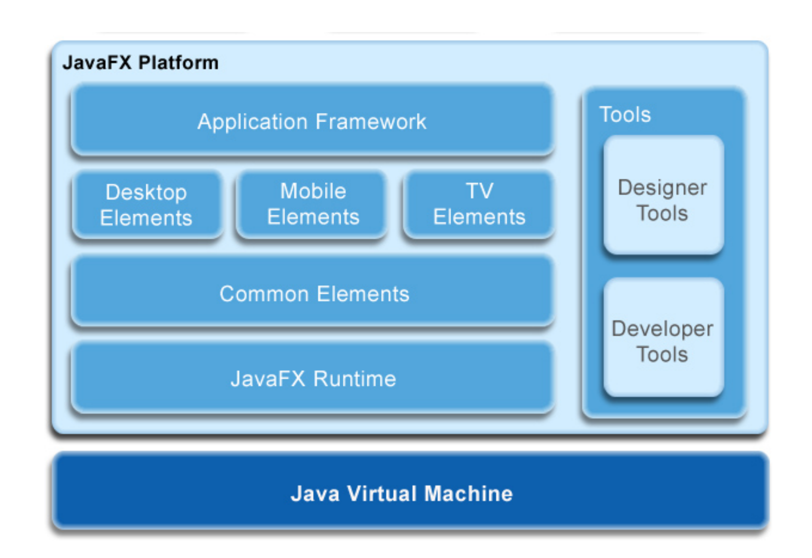


Abbildung 3.2: JavaFX Plattform[19]

[16] vgl. http://blogs.sun.com/javafx/entry/javafx_1_0_is_live, zugegriffen am 21. April 2009

[17] "Software as a Service" bezeichnet ein Geschäftsmodell, welches auf Dienstleistungen rund um Software basiert.

[18] vgl. http://www.pcwelt.de/start/software_os/online/news/189120/javafx_startet_offiziell_ins_rich_internet_rennen, zugegriffen am 21. April 2009

[19] vgl. http://www.javafx.com/about/overview, zugegriffen am 21. April 2009

Im Gegensatz zu Flex und Silverlight baut derjenige Teil, welcher die Oberfläche beschreibt, nicht auf XML bzw. einem XML-Derivat auf. Stattdessen wurde eine spezielle Skriptsprache namens “JavaFX Script” eingeführt. Im folgenden Code-Beispiel wird die Erstellung eines Buttons dargestellt.

```
1 Button { //Komponente
2   x:10 y:10 //Positionierung
3   text:"Klick mich!" //Beschriftung
4   action: operation() { //Funktion
5
6   }
7 }
```

Quellcode 3.2: Beispiel JavaFX-Button

In Abb. 3.2 wird die JavaFX Plattform veranschaulicht.

3.2.3 Fazit

Nachfolgend sollen die Aspekte Performance und Eindruck sowie technische Voraussetzung auf Anwender- und Entwicklerseite betrachtet werden. Abschließend wird der Einfluss der Open Source Philosophie diskutiert, da die in dieser Arbeit beschriebenen Technologien in wesentlichen Teilen darauf beruhen.
Die drei beschriebenen Produkte stellen die zur RIA-Entwicklung notwendigen Grundfunktionen zur Verfügung. Die Komplexität darauf aufbauender Funktionen wird bei keinem der drei Produkte konzeptionell begrenzt. Wie sich eine zunehmende Komplexität auf die Performance auswirkt, wurde im Rahmen dieser Recherche nicht untersucht. Der subjektive Eindruck des Autors ist, dass bei JavaFX die Darstellung stockend abläuft und die Ladezeiten selbst bei kleinen Applikationen ungewöhnlich lang sind. Dadurch kann angenommen werden, dass sich die Produkte in diesem Punkt unterscheiden.
Die im Vergleich zum Flashplayer und dem Java Plugin geringe Verbreitung des Browserplugins Silverlight (vgl. Abb. 3.3) führt dazu, dass der Endanwender in der Regel die Technik nicht ohne zusätzlichen Installationsaufwand nutzen kann. Die notwendigen Plugins sind für alle gängigen Browser kostenlos verfügbar, wobei diese Browser wiederum auf der Mehrzahl der Betriebssysteme lauffähig sind. Hierdurch ergibt sich eine de facto Plattformunabhängigkeit.
Auf Entwicklerseite ergeben sich, wenn auch nur geringe, Einstiegshürden aufgrund der neueingeführten Objektbeschreibungssprachen. Im Falle von JavaFX ist der initiale Aufwand, sich die Sprache JavaFX Script anzueignen, aufgrund ihrer Andersartigkeit (vgl. Quellcode 3.2) bezüglich bestehender Sprachen am größten.
Die technische Voraussetzung zur Entwicklung von RIAs mit der jeweiligen Technologie sind entsprechende Programmierwerkzeuge. Diese werden von den Herstellern im Wettbewerb um Marktanteile kostenlos zur Verfügung gestellt. Zusätzlich wird ihre Attraktivität gesteigert, indem möglichst viele Plattformen unterstützt werden. Des Weiteren festigt sich der Trend hin zu Open Source (Java, Flex, XML etc.). Mit dieser Philosophie gehen folgende gruppenspezifische Vorteile einher.

	Flex	Silverlight	JavaFX
Aktuelle Version	3.1	2.0	1.0
Sprachen	MXML ActionScript	XAML, JavaScript ASP.NET u. weitere	Java JavaFX Script
Browser benötigt	mind. Adobe Flash Player 9	Silverlight Plugin	Java Plugin mit JavaFX Erweiterung
Verbreitung des Plugins*	97%	21%	75%

* vgl. http://www.riastats.com (Stand März 2009)

Abbildung 3.3: RIA-Vergleich

- **Anwender**
  Der Anwender profitiert von einem guten Community-Support. Ihm wird freie und kostenlose Software zur Verfügung gestellt, welche in der Regel qualitativ hochwertig ist. Durch Offenlegung und Bereistellung des Quellcodes wird sichergestellt, dass Programme (auch zukünftig) prinzipiell durch einen großen Personenkreis weiter entwickelt werden können (u. A. Bilden eines Forks[20]). Im Allgemeinen wird bei der Entwicklung von Open Source Software großen Wert auf die Verwendung von Standards gelegt. Dadurch erhöht sich die Interoperabilität und die Abhängigkeit des Anwenders von einer bestimmten Software resp. eines Datenformats (MS Word) nimmt ab.

  Insgesamt führt die Existenz von alternativen Open Source Programmen, welche mit kommerzieller Software konkurrieren, zu mehr Wettbewerb um die Gunst des Kunden.

- **Entwickler und Contentanbieter**
  Alle die Anwender betreffenden Vorteile lassen sich auf Entwickler und Contentanbieter übertragen. Zusätzlich ermöglicht die Quelloffenheit zum Einen die einfache Anpassung des Programms an individuelle Bedürfnisse (Fork-Bildung), andererseits bieten frei verfügbare Software und Softwarekomponenten Zeit- und Kostenersparnis.

- **Software-Unternehmen**
  Im Vergleich zu den beiden vorher genannten Gruppen fällt die Liste der Vorteile für die Software-Unternehmen kürzer aus. Unternehmen profitieren von den freiwilligen Leistungen der Community in Bezug auf die Weiterentwicklung und Verbreitung von Software, Konzepten und Sprachen. Das Kerngeschäft verlagert sich teilweise von der Vermarktung eines Produkts hin zu

[20] Ein Fork bezeichnet in diesem Fall die unabhängige Entwicklung eines Programms, die auf dem Quellcode einer anderen Software aufbaut.

Dienstleitungen um ein Produkt (Consulting, d. h. Konzeption von Software-Lösungen) bzw. einer Sprache (kommerzielle Vermarktung von Entwicklungsumgebungen). Aufgrund der Vorteile, die sich für Anwender, Entwickler und Contentanbieter ergeben, steigt deren Akzeptanz gegenüber Unternehmen, die sich im Open Source Bereich engagieren.

Der einzige gravierende Nachteil welcher mit Open Source einher geht ist der für Software-Unternehmen zunehmende Wettbewerb.

3.3 Kommunikation zwischen Flex und PHP

In den folgenden Abschnitten werden verschiedene Konzepte der Kommunikation zwischen Flex und PHP erläutert. Von Relevanz sind diese Verbindungen deshalb, weil Datenbanken nicht direkt mittels Flex angesprochen werden können. Hierzu wird ein zusätzliches Backend benötigt, welches für das betrachtete Projekt bereits in der Sprache PHP realisiert ist (vgl. Abb. 3.4).

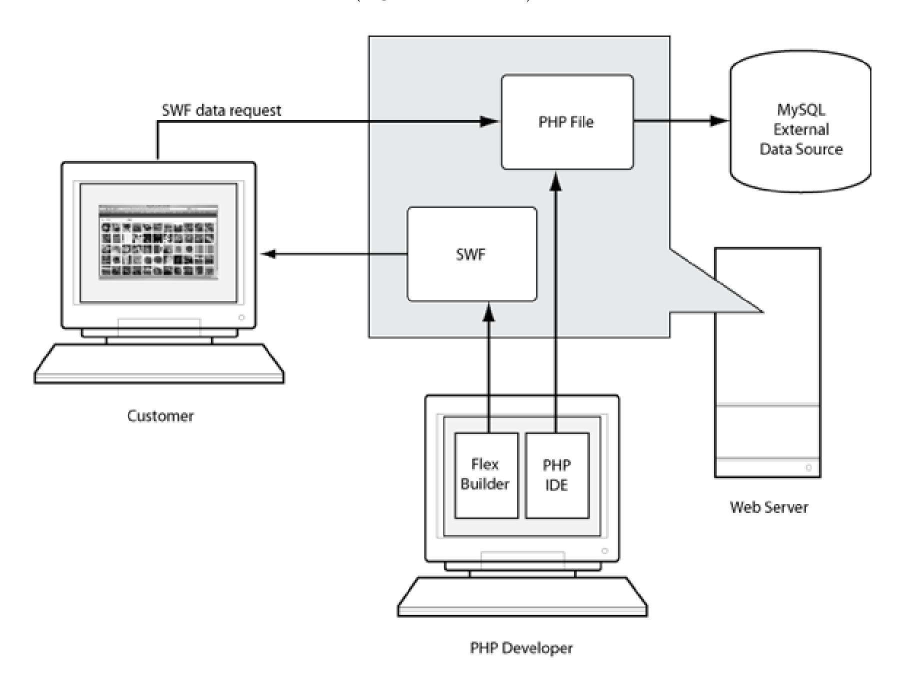


Abbildung 3.4: Integration von Flex und PHP[21]

Neben PHP erlaubt Adobe Flex die Verwendung weiterer Sprachen, welche auch das Ansprechen von Datenbanken ermöglichen. Zu den gängigsten zählen ASP.NET, Python sowie Java. Die Aufgabe besteht somit in der Entwicklung einer geeigneten Schnittstelle. Sie sollte den schnellen und sicheren Zugriff auf die PHP-Klassen

[21] vgl. `http://learn.adobe.com/wiki/display/Flex/Flex+and+PHP`, zugegriffen am 21. April 2009

gewährleisten. Eine weitere wichtige Funktion der Schnittstelle ist die Trennung zwischen datenverwaltendem resp. verarbeitendem Backend sowie der Darstellung. Wie bereits in Abschnitt 2.4 beschrieben, kommt in Flex mit ActionScript und MXML ein ähnliches Konzept zur Anwendung. Im optimalen Fall erfolgt eine so strikte Aufteilung, dass bei Änderungen auf PHP-Seite keine Modifikationen im Flex-Client nötig sind. Nachteilig ist sonst, neben dem zusätzlichen Programmieraufwand, das erforderliche Neukompilieren und Veröffentlichen der Flex-Applikation.
Ein wesentliches Merkmal der Kommunikation zwischen Flex und PHP ist, dass jegliche Anfragen stets vom Flex-Client an das PHP-Backend gerichtet werden. Der Grund hierfür ist, dass die Aktionen vom Anwender ausgehen und dieser nicht direkt auf das PHP-Backend zugreifen kann. Somit müssen die Anfragen über eine Verbindung vom Flex-Client an PHP weitergeleitet werden. Diese Verbindung kann mit unterschiedlichen Methoden aufgebaut werden. Bevor jedoch auf den konkreten Schnittstellenvergleich eingegangen wird, werden zunächst generelle Funktionsweisen unterschiedlicher Verbindungskonzepte zwischen Flex und PHP beschrieben.

3.3.1 Webservice

Webservices sind über das Internet abrufbare Dienstleistungen (z. B. Websuche, Warenlogistik). Um die Nutzung dieser Services so einfach wie möglich zu gestalten, setzen die Anbieter auf so genannte APIs[22]. Das sind auf offenen Standards basierende Schnittstellen, welche die Abhängigkeit von bestimmten Programmiersprachen und Betriebssystemen zwischen Anbieter und Nutzer vermeiden (vgl. [KW02]). Die Daten werden generell über das XML-Format ausgetauscht. Flex bietet für die Nutzung von Webservices die Komponente <mx:HTTPService> (vgl. Abs. 3.4.1) an.

3.3.2 Remoting und RPC

Remoting beschreibt das Senden von Objekten und Methodenaufrufen an andere Systeme. Im Gegensatz zu einem Webservice wird kein XML-Format gefordert. Flex unterstützt das Remoting durch die <mx:RemoteObject>-Komponente.
Die so genannten RPCs sind ein Bestandteil des Remotings. RPC steht dabei für "Remote Procedure Call" und bezeichnet eine Methode, um Daten zwischen Client und Server zu kommunizieren. Diese Technik wird verwendet, wenn die Prozedur, die aufgerufen werden soll, sich in einem anderen Adressraum befindet. So genannte Stubs[23] setzen die Zugriffe auf das RPC-Protokoll um (vgl. Abb. 3.5).

> "Im Stub erfolgen die Umsetzung auf eine netzwerktaugliche Darstellung und der Aufruf des RPC-Protokolls."[24]

[22] API ist die Abkürzung für "Application Programming Interface" und bezeichnet eine Programmierschnittstelle.
[23] engl. für Stumpf
[24] vgl. `http://www.microsoft.com/germany/technet/datenbank/articles/600855.mspx#EYB`, zugegriffen am 23. April 2009

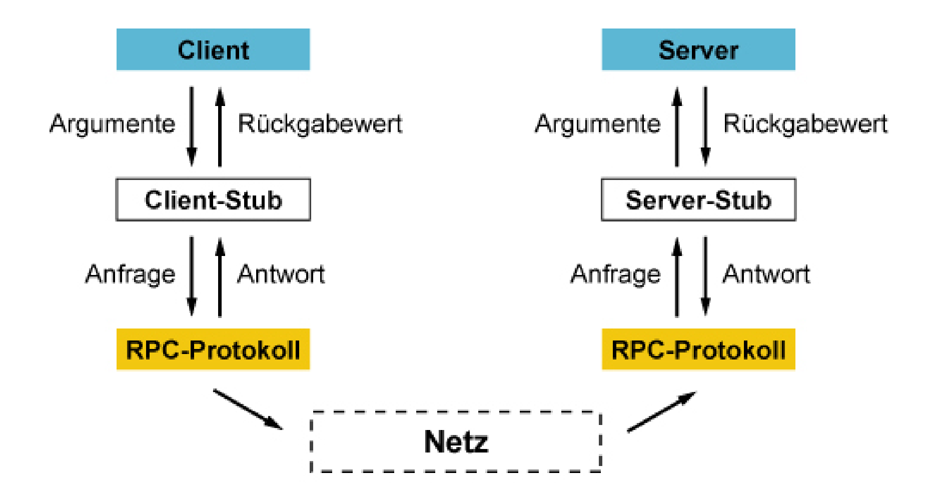


Abbildung 3.5: Ablauf eines RPCs

Es wird eine vorher definierte Methode (<mx:method>) des lokalen Remote-Objekts (<mx:RemoteObject>) aufgerufen, die das Ergebnis über die im Remote-Objekt angegebene Rückruffunktion[25] übermittelt. Es ist also keine Kenntnis über die spezifische Art, wie die Daten gesendet und empfangen werden, notwendig. Die Details der RPC-Implementierung sind so abstrahiert, dass es für den Programmierer wie ein Aufruf einer lokalen Funktion wirkt.

3.3.3 Serialisierung und Deserialisierung

Die beschriebenen Remote Procedure Calls erfordern das so genannte Serialisieren. Dabei werden Objekte in Bits oder auch XML-Daten konvertiert, um diese speichern oder in einem Netzwerk übermitteln zu können. Die zurückgegebenen Daten sind vor ihrer Weiterverarbeitung zunächst durch Deserialisieren wieder in Objekte umzuwandeln. Die binäre Übermittlung wird mittels dem Action Message Format (AMF), das auch im Zend Framework mit dem Paketnamen "Zend_Amf" (siehe Abs. 3.4.4) implementiert wurde, unterstützt. Auf das genauere Verhalten wird im nächsten Abschnitt eingegangen.

3.3.4 Action Message Format (AMF)

Beim Action Message Format handelt es sich um ein binäres Format, welches die Serialisierung und Deserialisierung von ActionScript-Objekten übernimmt.

[25]Bei einer Rückruffunktion (auch 'callback function' genannt) handelt es sich um eine Funktion, die als Parameter einer anderen Funktion übergeben und unter entsprechenden Voraussetzungen von dieser auch aufgerufen wird.

> "AMF allows for native data types and complex object mapping between the client and the server. Enabling ActionScript and PHP developers to not worry about type casting between the two languages."[26]

Dadurch können Client und Server unterschiedliche Datentypen, wie z. B. Arrays oder selbstdefinierte Objekte, typsicher austauschen. Die erste Version von AMF (AMF0) unterstützt die Versendung von Objekten, indem lediglich die Referenz übermittelt wird. Dadurch werden redundante Instanzen vermieden. Objektbeziehungen können wiederhergestellt werden und Zirkelbezüge[27] sind umsetzbar, ohne endlose Rekursionen[28] bei der Serialisierung zu erwarten. Die aktuelle Version AMF3 ist Open Source und bietet zusätzlich die Möglichkeit, lediglich die Eigenschaften eines Objekts zu verschicken. Außerdem werden neuere Datentypen, welche in ActionScript 3.0 eingeführt wurden, nun auch unterstützt[29].

3.3.5 Representational State Transfer (REST)

Webseiten verwenden in der Regel Cookies oder Session-IDs (vgl. Abs. 4.6.3) um Zustände zu speichern. Ein anderer Ansatz wird mit der REST-Architektur verfolgt. Das Akronym steht dabei für "Representational State Transfer". Die Zustände vorheriger Botschaften werden weder server- noch clientseitig gespeichert. Alle für die Übermittlung notwendigen Informationen wie z. B. auch Authentifizierungsdaten sind in jeder einzelnen Nachricht enthalten. Einerseits erhöht dieses Konzept bei vielen unterschiedlichen Anfragen eines Clients die Datenlast. Andererseits wird durch eine REST-konforme Umsetzung ein einfaches "Load Balancing"[30] ermöglicht (vgl. [RR07]), welches wiederum die Performance erhöhen kann.
Bei der Umsetzung mit Hilfe von Flex können sich jedoch Schwierigkeiten ergeben, wie das folgende Zitat belegt.

> "The current Flex HTTPService component is severely lacking when it comes to making REST calls. This component can only send GET and POST requests (no PUT, DELETE, HEAD, or OPTIONS without using a proxy) and does not make available the HTTP status code returned in the response. Rather, it fires a fault event for any HTTP response that does not have a status code of 200 (even a 201 "created" status is considered a fault condition)."[31]

Somit reichen die derzeit implementierten Funktionen der HTTPService Komponente nicht aus, um mit der REST-Architektur eine voll funktionierende Schnittstelle über HTTP zu PHP aufzubauen.

---
[26]vgl. `http://amfphp.sourceforge.net`, zugegriffen am 24. März 2009
[27]Ein Zirkelbezug entsteht, falls voneinander abhängige Objekte sich gegenseitig referenzieren.
[28]Eine Rekursion bezeichnet eine sich selbst aufrufende Funktion.
[29]vgl. `http://osflash.org/documentation/amf3`, zugegriffen am 10. April 2009
[30]In diesem Fall bezeichnet das Load Balancing die Lastverteilung auf parallel arbeitende Server.
[31]vgl. `http://code.google.com/p/resthttpservice`, zugegriffen am 27. März 2009

3.4 Schnittstellenvergleich

In diesem Abschnitt werden konkrete Methoden vorgestellt, welche die Kommunikation zwischen PHP und Flex ermöglichen. Nachdem die Möglichkeiten beschrieben sind, wird abschließend ein Fazit gezogen.

3.4.1 HTTPService

Eine schnell und einfach umsetzbare Variante, Daten aus einem PHP-Backend zu erhalten, ist die Nutzung der in Flex implementierten <mx:HTTPService>-Komponente. Diese verwendet lediglich das HTTP-Protokoll und benötigt somit keine weiteren Implementierungen wie z. B. ein Gateway[32]. Es können unterschiedliche Text-Strukturen, meist handelt es sich dabei um XML, (auf die unterschiedlichen XML-Formatierungen soll hier jedoch nicht näher eingegangen werden) übermittelt werden [33]. Die Abwicklung verläuft über ein in der MXML-Datei erstelltes HTTPService-Objekt (<mx:HTTPService>). Es verfügt über die Methode `send()`, die beim Aufruf einen HTTP-Request mit gewünschten Parametern zu der im Objekt angegebenen Zieladresse schickt. Ohne Verwendung eines Proxies[34] (mit Proxy-Nutzung ergeben sich weitere Möglichkeiten, sind jedoch hier nicht relevant) können ausschließlich die HTTP-Methoden GET und POST[35] verwendet werden. Damit wird auf PHP-Seite bereits ermöglicht, Datenbankabfragen auszuführen, das Ergebnis in XML-Struktur umzuwandeln und die Daten per HTTP-Antwort zurückzugeben[36]. Das Ergebnis wird in dem Objekt `event.result` gespeichert, welches noch einer Typkonvertierung bedarf. Diese lässt sich im Vorfeld durch den Parameter `resultFormat` (vgl. Quellcode 3.3) in dem HTTPService-Objekt bestimmen.

```
1 <mx:HTTPService id="httpServiceObjekt" url="{phpDatei}" method="
    POST" result="handleResult(event)" fault="handleFault(event)"
    resultFormat="text">
2   <mx:request>
3     <username>{username.text}</username>
4   </mx:request>
5 </mx:HTTPService>
```

Quellcode 3.3: Beispiel HTTPService Objekt

Da das PHP-Backend in diesem Projekt bereits vorhanden ist und die Rückgabewerte nicht im XML-Format vorliegen, handelt es sich bei dieser Umsetzung um einen erheblichen Mehraufwand. Alle benötigten Daten im PHP-Backend müssten in eine XML-Struktur integriert werden, um diese an den Flex-Client übermitteln

[32] Gateway bezeichnet in diesem Fall eine protokollunabhängige Vermittlung.
[33] vgl. `http://livedocs.adobe.com/flex/3/html/help.html?content=data_access_2.html`, zugegriffen am 27. März 2009
[34] Ein Proxy stellt einen Vermittler dar, der Anfragen entgegennimmt, um diese über seine eigene Adresse weiterzuleiten.
[35] GET und POST sind Methoden für HTTP, um einer Webseite Informationen zu senden.
[36] vgl. `http://www.adobe.com/devnet/flex/articles/php_getstarted_04.html`, zugegriffen am 22. Februar 2009

zu können. Die HTTPService-Komponente eignet sich somit insbesondere nur für die Übertragung kleiner Datenmengen, da die XML-Struktur viel Text erfordert und dadurch eine hohe Netzwerklast entstehen kann.

3.4.2 Simple Object Access Protocol (SOAP)

Bei SOAP handelt es sich um ein Datenaustauschformat. Seine Spezifikationen sind auf der offiziellen Webseite wie folgt beschrieben.

> "SOAP Version 1.2 (SOAP) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics."[37]

SOAP realisiert die Datenübertragung im XML-Format. In der Regel werden die Daten über HTTP gesendet, jedoch sind weitere Protokolle denkbar wie z. B. SMTP. Diese Untersuchung bezieht sich dabei nur auf das Protokoll HTTP.

> "Eine SOAP-Nachricht ist im Grunde eine "one-way"-Übertragung zwischen SOAP-Knoten, von einem SOAP-Sender zu einem SOAP-Empfänger. Allerdings können Applikationen SOAP-Nachrichten so kombinieren, dass vielschichtigere Interaktionsformen, von einer einfachen Anfrage/Antwort-Kommunikation bis hin zum multiplen, wechselseitigen Nachrichtenaustausch, realisierbar sind."[38]

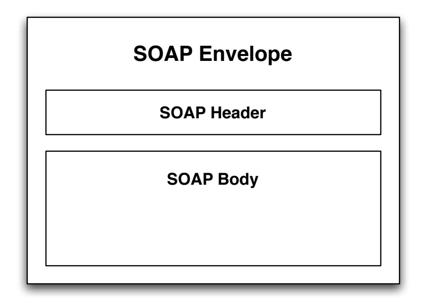


Abbildung 3.6: Aufbau einer SOAP-Nachricht

Der Aufbau einer SOAP-Nachricht wird in Abb. 3.6 schematisch dargestellt.

[37] vgl. `http://www.w3.org/TR/soap12-part1`, zugegriffen am 26. Februar 2009
[38] vgl. `http://poseidon.home.tlink.de/w3c/REC-soap12-part0-20030624-de_DE/#L1161`, zugegriffen am 26. Februar 2009

Ebenso wie bei der HTTPService-Komponente erfolgt die Datenübertragung bei der Verwendung des XML-Formats textuell. Im Vergleich zur binären Übertragung führt diese zu einer wesentlich höheren Netzwerklast. Daher soll in den nächsten beiden Abschnitten eine Alternative zum XML-Format erörtert werden.

3.4.3 AMFPHP

AMFPHP setzt sich aus den Abkürzungen AMF (vgl. Abs. 3.3.4) und PHP (vgl. Abs. 2.1) zusammen.

> “AMFPHP was the first open-source Remoting gateway and continues to be developed.”[39]

Es ist ein Instrument, das den Datenaustausch zwischen ActionScript und PHP mittels AMF realisiert. Dieser geschieht, im Gegensatz zu den bereits vorgestellten Übertragungstechniken, binär. Serialisierung und Deserialisierung werden von AMFPHP im Hintergrund automatisch durchgeführt.

> “AMFPHP allows thin client applications built in languages such as Flash, Flex, and AIR to communicate directly with PHP class objects on the server.”[40]

Zusätzlich zu Objekten und Arrays können auch Rückgabewerte von Datenbankanfragen übertragen werden.

3.4.4 Zend_Amf

Zend_Amf ist ein Paket des Zend Frameworks. Ebenso wie AMFPHP handelt es sich dabei um eine von Wade Arnold entwickelte Schnittstelle. Sowohl Zend_Amf wie auch AMFPHP ermöglichen eine effiziente binäre Datenübertragung zwischen ActionScript und PHP.

> “However there will be one major change the core of AMFPHP will be built from the Zend Framework AMF library which is being maintained be myself and the rest of the Zend Framework community. AMFPHP will have an additional code base that continues to make it simple for beginner to intermediate developers to start working. That is really the only difference in functionality.
> Let‘s face it if you were trying to use AMFPHP in an enterprise capacity it did not meet your needs.”[41]

Laut dem führenden Entwickler von AMFPHP ist dieses für kleinere Projekte konzipiert und eignet sich aufgrund der einfachen Anwendung besonders für den Einstieg

[39]vgl. `http://osflash.org/projects/amfphp`, zugegriffen am 23. März 2009
[40]vgl. `http://amfphp.sourceforge.net`, zugegriffen am 23. März 2009
[41]vgl. `http://wadearnold.com/blog/?p=112`, zugegriffen am 8. April 2009

in diese Thematik.
Vor der Zend Framework Version 1.7 war das Zend_Amf-Paket separat als funktionale Erweiterung zum Framework erhältlich. In jüngster Zeit konnte sich das Zend_Amf-Paket gut etablieren und ist mittlerweile fester Bestandteil des Frameworks. Um lediglich Zend_Amf ohne das komplette Framework nutzen zu können, werden die Pakete "Zend_Amf" und "Zend_Server" sowie die "Exception.php" benötigt.
Es sei an dieser Stelle darauf hingewiesen, dass sich für den Einstieg in Zend_Amf eine von Richard Bates erstellte Hilfsapplikation "CRUDdy Buddy"[42] gut eignet. Diese ist in AIR geschrieben und generiert PHP-, ActionScript- und MXML-Code um Flex mit PHP über Zend_Amf zu verbinden.

3.4.5 Vergleichsanalyse

In diesem Abschnitt werden die Ergebnisse eines Performancevergleichs einiger zuvor beschriebener Verbindungsoptionen präsentiert. Die gezeigten Ergebnisse wurden mit Hilfe einer von James Ward geschrieben Flex-Applikation namens "Census"[43] generiert.

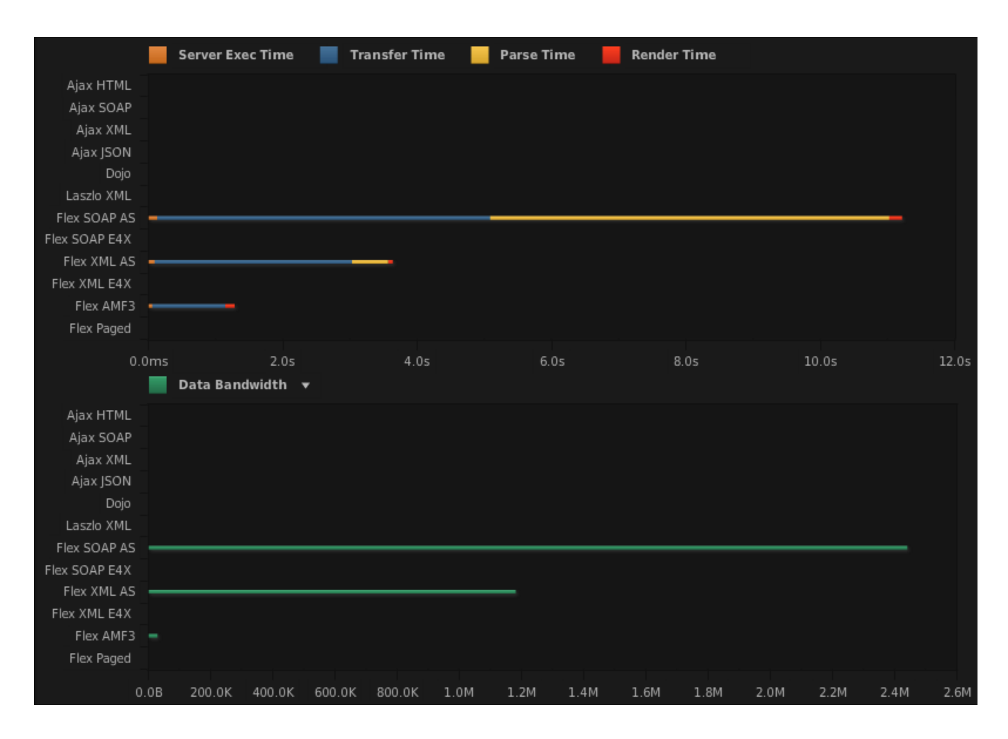


Abbildung 3.7: RIA Data Benchmark

[42]Der Teil "CRUD" steht für "Create, Retrieve, Update, Delete", siehe `http://code.google.com/p/cruddybuddy`
[43]siehe `http://www.jamesward.com/census`

In Abb. 3.7 wird ein Vergleich der für diese Untersuchung relevanten Schnittstellen dargestellt. Konkret handelt es sich dabei um SOAP, XML und AMF3. Gemessen wird neben der auftretenden Netzwerklast auch die benötigte Übertragungszeit für die Übermittlung von 5000 Datenbankzeilen. Die Daten werden im Flex-Client als ActionScript-Objekte benötigt. Im Performance-Test werden die Übertragungszeiten in folgende Elemente untergliedert:

- **Server Exec Time**
  Die Server Execution Time spiegelt die Verarbeitungszeit auf dem Server wider. In allen drei Testfällen ergeben sich keine nennenswerten Unterschiede. Falls für die Übertragung GZip[44] verwendet wird, müssen die Daten auf dem Server zunächst komprimiert werden. Dieses Vorgehen reduziert einerseits das zu übertragende Datenvolumen, wodurch die benötigte Übertragungszeit ("Transfer Time") verringert wird. Andererseits erhöht sich die Serverlast durch das Ausführen der Kompressionsalgorithmen. Ob sich insgesamt eine Beschleunigung durch GZip erreichen lässt, hängt stark von der Beschaffenheit der Daten ab.

- **Transfer Time**
  Die Transfer Time gibt die eigentliche Übertragungszeit vom Server zum Client an. Die Dauer der Übermittlung unterscheidet sich aufgrund der Datengröße. Binär kodierte Daten benötigen in der Regel weniger Speicherplatz als solche im XML-Format. Aus diesem Grund erlaubt AMF3 die schnellste Datenübermittlung.

- **Parse Time**
  Mit Parse Time ist die Dauer der Datenumwandlung gemeint. Durch die Verwendung von AMF3 erfolgt die Datenübertragung typsicher. Somit werden ActionScript Objekte übermittelt, wodurch keine Typkonvertierung erforderlich ist. Dadurch ist die Parse Time gegenüber der Gesamtübertragungszeit marginal.

- **Render Time**
  Die Datenbankzeilen werden mit Hilfe der Komponente <mx:DataGrid> angezeigt. Hierfür sind die übertragenen Daten zu rendern. Diese Aufgabe geschieht lokal im Client, weshalb die Darstellungsgeschwindigkeit u. A. von der Leistung des Rechners abhängt.

- **Data Bandwidth**
  Die Übertragung mittels AMF3 erzeugt eindeutig die geringste Bandbreite. Der Grund hierfür liegt in der binären Kodierung der Daten, welche wesentlich speichereffizienter ist als das XML-Format.

Der Vergleich verdeutlicht, dass AMF3 sowohl bezüglich der Bandbreite als auch der Übertragungsgeschwindigkeit die beste Performance besitzt. Durch die kurze

[44]Bei GZip handelt es sich um ein Programm für die Datenkompression.

Verarbeitungszeit auf Serverseite ("Server Execution Time") sowie der sehr geringen Parse Time konnten mit AMF3 5000 Datenbankzeilen innerhalb von 1,2 Sekunden übermittelt werden. Beim Einsatz von SOAP ist der zeitliche Aufwand ca. zehn mal höher.
Es sprechen somit mehrere Gründe für die Verwendung von AMF3. Da das PHP-Backend bereits auf dem aktuellen Zend Framework aufbaut, bietet sich die Verwendung des integrierten Pakets Zend_Amf an.

3.5 Fazit

Nach der Analyse der Schnittstellen wird deutlich, dass das PHP-Backend zu erweitern ist, um die Verbindung mit Flex herstellen zu können. Über die Schnittstelle kann kein ausführbarer Code übertragen werden. Daher müssen auf PHP-Seite alle für das Reporting relevanten Daten als Eigenschaften in einem Objekt zu Verfügung gestellt werden. Die zu übertragenden Objekte werden allgemein als "ValueObjects" bezeichnet und mit "VO" abgekürzt. Diese Abkürzung wird bei der Namensgebung von Klassen in der Regel als Suffix (z. B. ReportingVO) verwendet. Die Flex-Applikation benötigt zusätzlich eigene VO-Klassen, welche die gleichen Eigenschaftsnamen besitzen wie die VOs auf PHP-Seite. Nachdem diese Architektur implementiert ist, können die Daten durch so genanntes "Class Mapping" übermittelt werden. Dabei werden die Daten bereits auf PHP-Seite in einer geeigneten Form in Objekten abgelegt, so dass sie direkt vom Flex-Client verwendet werden können. Der prinzipielle Ablauf ist in Abb. 3.8 skizziert.

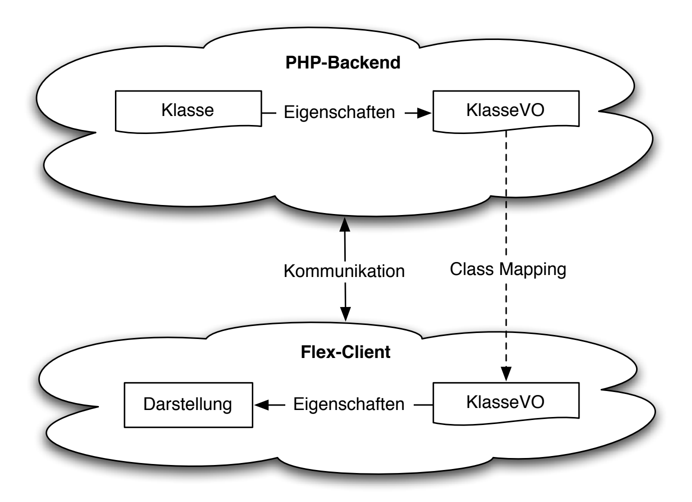


Abbildung 3.8: Class Mapping

Die ValueObjects können sowohl empfangen als auch versendet werden. Für das Reporting ist nur der Empfang solcher VOs relevant.

4 Konzeption

Im Anschluss an die Untersuchungen wird ein Konzept zur Realisierung des Reportingclients in Form einer RIA-Applikation entwickelt. In diesem Kapitel wird auf spezielle Funktionen des Reportingclients eingegangen, weshalb auch einige Varianten des E-Mail-Reportings zu erörtern sind. Anhand dieser konkreten Problemstellung werden allgemeine Probleme bei der Erstellung von RIA erläutert.
Das bisherige Reporting hat, wie in der Einleitung und Motivation erwähnt wurde, einige Unzulänglichkeiten, welche im neuen Reporting behoben werden sollen. Zudem sind die alten Funktionalitäten um weitere zu ergänzen.
Im Gegensatz zur konventionellen Softwareentwicklung kann es beim Einsatz von Flex sinnvoll sein, zuerst die Oberfläche zu gestalten und anschließend die Funktionen zu implementieren. Das betrachtete Softwareprojekt stellt hierfür ein gutes Beispiel dar. Hier wurde zunächst die Oberfläche entworfen, um sie als Prototyp für Usability-Tests zu nutzen. Zum einen erlauben solche Tests ein Feedback durch mehrere Probanden (andere Entwickler, ggf. Kunden), wodurch Unzulänglichkeiten in einer sehr frühen Phase der Entwicklung identifiziert werden können. Aus Gründen der Usability sollten die zu erwartenden Benutzerinteraktionen (Werteeingabe, Betätigung von Buttons etc.) die Oberflächenstruktur bestimmen und nicht umgekehrt. Zudem ist die Mehrzahl der bestimmenden Variablen in den Objekten definiert, aus denen sich die Oberfläche zusammensetzt. Das bedeutet, dass mit der Aggregation der Oberfläche die Variablendefinition automatisch durch die Flex-Umgebung erfolgt.
In Abb. 4.1 wird der Projektrahmen skizziert um zu verdeutlichen, welche Bereiche bereits implementiert und welche noch zu realisieren sind.
Das PHP-Backend ist um die Remote-Klassen zu erweitern. Diese stellen die Reportingdaten für die Flex-Applikation zur Verfügung. In den folgenden Unterkapiteln wird auf mögliche technische Umsetzungen eingegangen.

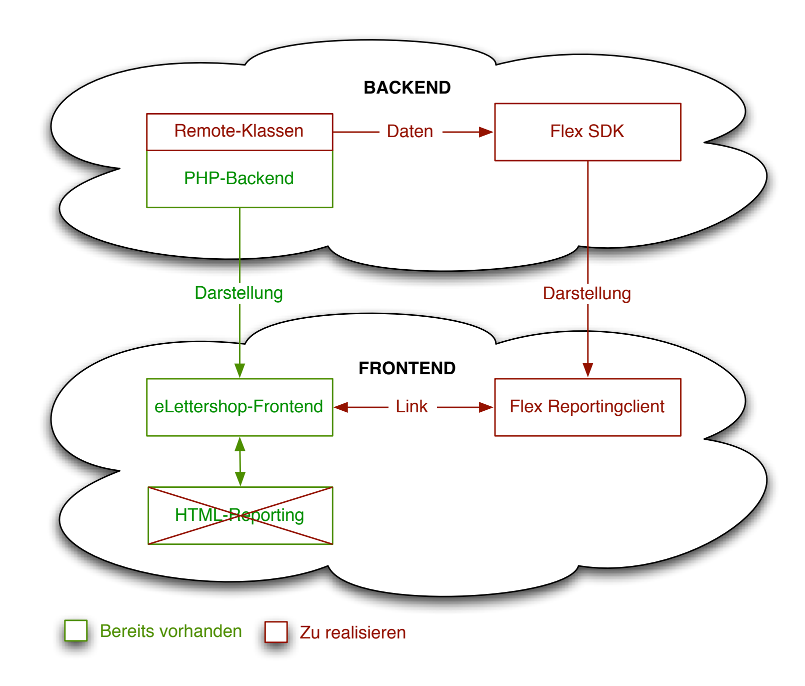


Abbildung 4.1: Projektskizzierung

4.1 Grundfunktionen des Reportings

In den folgende Abschnitten werden die Grundfunktionen der Reporting-Applikation beschrieben. Das Portfolio dieser Funktionen ist maßgeblich durch Kundenwünsche geprägt worden.

4.1.1 Darstellung von Mailings

Aufgrund der teilweise hohen Anzahl im System befindlicher, kundenspezifischer Mailings wird eine geeignete Darstellungsform benötigt. Diese soll im neuen Reportingclient effizienter, übersichtlicher, flexibler und intuitiver gestaltet sein. Hierzu bietet sich die Darstellung als Baumstruktur an, welche eine einfache Abbildung hierarchischer Strukturen erlaubt (vgl. Abb. 4.2). Für die Kunden ist zudem der Vergleich von thematisch ähnlichen Mailings von Interesse. Das bestehende Tool bietet jedoch lediglich eine nach Versanddatum sortierte Auflistung. Somit ist eine entsprechende Gruppierungsfunktion umzusetzen.

Um indiviudelle Ansichten zu gewährleisten, wird ein multifunktionaler Filter implementiert. Dieser listet nur die Mailings auf, welche die angegebenen Kriterien erfüllen. Näheres zur Filterung wird im nächsten Abschnitt erläutert.

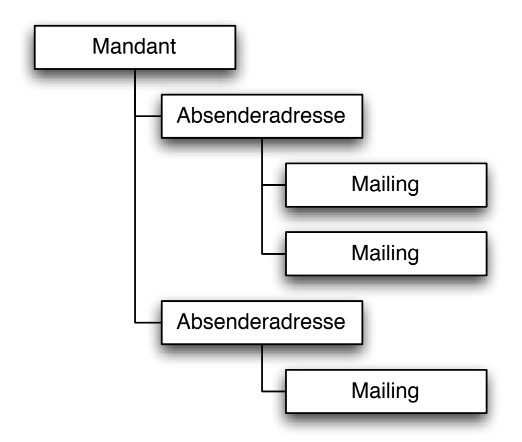


Abbildung 4.2: Baumstruktur

4.1.2 Filterung der Mailings

Damit der Benutzer schnell und unkompliziert das gewünschte Mailing in der Baumstruktur lokalisieren kann, ist neben einer direkten Suchfunktion ein Filter erforderlich. Suchfunktionen nehmen Suchwörter als Argument entgegen und geben als Ergebnis alle Datensätze zurück, welche die übergebenen Begriffe enthalten. Filter hingegen erlauben es, die Suchergebnisse weiter einzugrenzen, d. h. die Suche weiter zu spezifizieren.
In der folgenden Liste werden Kriterien, nach denen gefiltert werden kann sowie die dafür geeigneten Darstellungskomponenten beschrieben.

- **Versanddatum**
  Der Filter für das Versanddatum erlaubt die Angabe eines Zeitfensters, dessen Beginn und Ende über zwei Datumsfelder zu definieren sind. Alle Mailings, deren Versendungsdatum in diesen Zeitraum fallen, werden entsprechend gruppiert.
  Das Flex SDK bietet eine Komponente an, die die Datumsauswahl über einen Kalender im Tool-Tip-Format ermöglicht. Bei der Übermittlung des Datums von Flex zu PHP muss das Format beachtet werden. Es ist generell sinnvoll, eine Konvertierungsfunktion für die Datumsformate zu programmieren, so dass durch einen Funktionsaufruf das benötigte Format zurückgegeben wird. Flex kann wie auch PHP ein Datum mit der Unixzeit[1] darstellen. Hierbei ist zu berücksichtigen, dass Flex intern in Millisekunden rechnet, PHP hingegen in Sekunden.

- **Status**
  Neben der zeitlichen Eingrenzung, ist eine Filterung nach dem Status pra-

[1] Die Unixzeit gibt ein Datum in Form von vergangenen Sekunden seit dem 1. Januar 1970 an (siehe `http://www.unixtimestamp.com`).

xisrelevant. Mit dieser Funktionalität kann sich der Kunde zum Beispiel alle aktiven, d. h. in der Bearbeitung befindlichen Mailings übersichtlich anzeigen lassen. Aus diesem Grund wird ein Statusfilter in Form einer Combobox, welche alle relevanten Status enthält, implementiert. Im Gegensatz zu einem Frei-Text-Feld, sieht der Anwender in der Combobox die wählbaren Status, so dass die Gedächtnisbelastung minimiert wird. Um eine Mehrfachauswahl zu ermöglichen, kann auch eine Auswahlliste (<mx:List> mit Parametereinstellung `allowMultipleSelection="true"`) als Komponente in Betracht gezogen werden. Der Nachteil liegt hierbei jedoch in dem zusätzlich benötigten Platz (vgl. Abb. 4.3) für jede extra angezeigte Zeile.



Abbildung 4.3: Combobox und Liste

- **Versendungstyp**
  Im PHP-Backend wird zwischen Live- und Testmailing unterschieden. Testmailings werden in der Regel an eine kleine Empfängerzahl versendet, um z. B. die Darstellung zu überprüfen. Der Anwender soll im Reportingclient die Möglichkeit haben, lediglich Live- oder Testmailings zu betrachten.
- **Mandant**
  Es gibt Nutzer, welche für ihre eigenen Kunden Mailings im eLettershop erstellen. In diesem Fall ist der direkte eLettershop-Kunde ein Mandant. Es bietet sich erneut eine Combobox an, da die Menge an Mandanten meist überschaubar ist. Eine Auswahl geschieht so schneller als die Suche über Mandanten-ID oder Mandantenname.

Der Ablauf der Filterung wird in Abb. 4.4 verdeutlicht. Alle gesetzten Filterkriterien werden als Parameter über ein Remote-Objekt an das PHP-Backend gesendet. Dort können diese verwendet werden, um die gewünschten Mailings als VOs in einem Array an den Flex-Client zu übermitteln. Anschließend werden die MailingVOs in der Baumstruktur abgebildet.

4.1.3 Status der Versendung

Dieser Status, im eLettershop "Response-Übersicht" genannt, spiegelt den aktuellen Stand der verschickten E-Mails wider. So werden z. B. bereits geöffnete E-Mails ausgewiesen. Falls sich das Mailing noch im Versendungsprozess befindet, lässt sich anhand eines Kreisdiagramms ablesen, wie viele E-Mails bereits verschickt wurden bzw. noch zu versenden sind. Verschickte E-Mails lassen sich wie folgt kategorisieren:

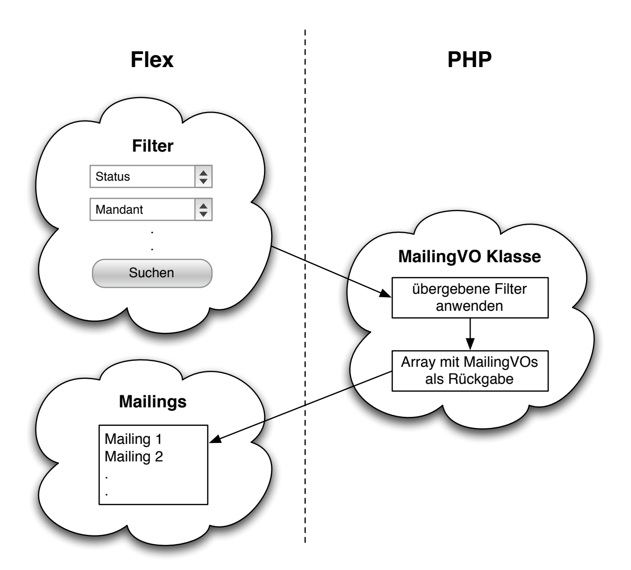


Abbildung 4.4: Ablauf der Filterung

- E-Mail geöffnet
- Mindestens ein Link wurde angeklickt
- SoftBounce
- HardBounce
- Noch keine Antwort erhalten

E-Mails, in denen ein Link geklickt wurde, stellen eine Submenge der Kategorie "E-Mail geöffnet" dar, welche gesondert ausgewiesen wird, ohne zu der Obermenge hinzugezählt zu werden. Auf diese Weise bietet sich dem Kunden eine detaillierte Response-Übersicht. E-Mails in der Kategorie "Keine Antwort" sind übermittelt, jedoch wurde noch keine Reaktion des Empfängers registriert. Genauere Information zu den beiden aufgelisteten Bounce-Arten werden in Abschnitt 2.6.3 gegeben.
Durch die Darstellung als Kreisdiagramm lässt sich die prozentuale Verteilung gut erkennen. Eine vereinfachte Form ist in Abb. 4.5 dargestellt. Im Reportingclient werden die einzelnen Segmente eingefärbt. Hierdurch heben sie sich besser voneinander ab. Zudem wird der Vergleich einzelner Response-Arten unterschiedlicher Mailings erleichtert. Die Relativ- und Absolutwerte können aus der Größe der Segmentflächen nur unpräzise resp. gar nicht abgelesen werden. Daher wird zusätzlich zur Einfärbung jedes Segment mit zwei Zahlenwerten versehen, welche eine absolute bzw. relative Angabe zu den Responses macht.

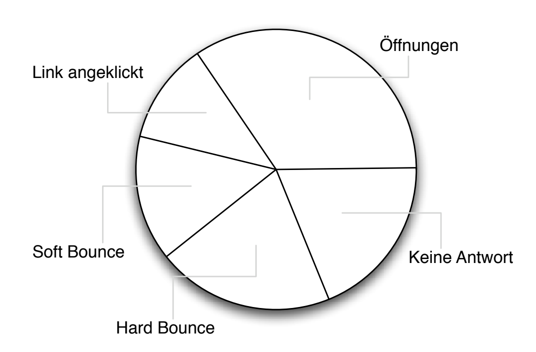


Abbildung 4.5: Kreisdiagramm

4.1.4 Klicks pro Stunde

Im eLettershop werden die Klicks auf Links in zwei unterschiedlichen Varianten dargestellt. Beide visualisieren die registrierten Klicks pro Stunde. Die folgende Auflistung nimmt eine Abgrenzung der Varianten vor.

- **Prozentualer Anteil der Klicks pro Stunde**
  In dieser Reportingfunktion wird die prozentuale Verteilung der Klicks pro Stunde visualisiert. So kann der Anwender sich darüber informieren, welcher Anteil der Gesamtklicks in welcher Stunde aufgetreten ist.
  Hierfür eignet sich ein Liniendiagramm, da so die Sättigung als Verlauf gut dargestellt werden kann.

- **Anzahl der Klicks pro Stunde**
  Bei diesem Diagramm werden die Klicks pro Stunde absolut ausgewiesen. Für diese einzelne Ausweisung bietet sich ein Balkendiagramm an, um die Verteilung auf die jeweilige Stunde hervorzuheben.

4.1.5 Linkauswertung

Neben der allgemeinen Klickauswertung sollen die einzelnen Links der E-Mail aufgelistet und ausgewertet werden. Die folgenden Kategorien beziehen sich jeweils nur auf einen speziellen Link im E-Mail Body:

- **Klicks gesamt**
  Die gesamten Klicks spiegeln alle Klicks der Empfänger wider. Es werden auch mehrere Klicks eines Empfängers auf einen Link gezählt.

- **Klicks eindeutig**
  Aus der Anzahl der eindeutigen Klicks wird nur ein Klick pro Empfänger gezählt. Mehrere Klicks eines Empfängers auf den gleichen Link bleiben hier somit unbeachtet.

- **Klickrate empfangene E-Mails**
  Die Klickrate gibt in diesem Fall an, in welchem Verhältnis die Klicks zu allen empfangenen E-Mails stehen.

$$Klickrate = \frac{Klicks}{Empfangene\ Mails} \tag{4.1}$$

- **Klickrate geöffnete E-Mails**
  Bei dieser Klickrate wird lediglich das Verhältnis der Klicks zu allen geöffneten E-Mails berechnet.

4.1.6 Diagramme anzeigen und vergleichen

Grundsätzlich werden für die Anzeige sowie den Vergleich von Diagrammen drei Bereiche benötigt;

- Auflistung der Mailings
- Auflistung der Funktionen
- Visualisierung der Ergebnisse

An dieser Stelle ist die Berücksichtigung der Usability sehr wichtig, da die zu gestaltende Oberfläche die wesentliche Benutzerschnittstelle darstellt. Sie soll kompakt, intuitiv und flexibel sein.
Im einfachsten Fall wird nur eine Reportingfunktion auf nur ein einzelnes Mailing angewendet. Intuitiv fängt der Ablauf mit der Auswahl eines Mailings an. Bevor die gewünschten Daten angezeigt werden, wird noch die dafür vorgesehene Reportingfunktion ausgewählt. Die Mailings besitzen abhängig von ihrem Status unterschiedliche Reportingfunktionen. Nur die, bei denen Tracking aktiviert ist und der Status sich in "SHIPMENT_RUNNING" oder "SHIPMENT_FINISHED" befindet, bieten alle Reportingfunktionen an. Bei der Mailing- und Funktionsauswahl handelt es sich jeweils um eine Auflistung. Da der Funktionsumfang vom Mailing abhängig ist, bietet es sich an, die Funktionen als Äste des Mailing-Knotens in der Baumstruktur anzuzeigen (vgl. Abb. 4.6). So werden lediglich die auf das Mailing anwendbaren Funktionen angezeigt wodurch der Anwender vor Fehlbedienungen bewahrt sowie eine gute visuelle Ergonomie erreicht wird.
Ein entscheidender Aspekt beim Anzeigen und Vergleichen von Diagrammen ist die Visualisierung der Ergebnisse. In Abschnitt 4.7.2 wird ein Konzept vorgestellt, das dem Anwender eine individualisierbare Fläche bietet. Neben dem Vergleich einzelner Diagramme, die sich in getrennten Feldern befinden, soll der Nutzer die Möglichkeit haben, eine Reportingfunktion für mehrere Mailings in einer gemeinsamen Darstellung anzuzeigen. Diese Option ist lediglich bei der Graphenansicht relevant. Die Charting-Komponenten in Flex erlauben das Plotten mehrerer Graphen innerhalb eines Diagramms, was den direkten Vergleich der Datensätze ermöglicht. Das Hinzufügen weiterer Datensätze zu einem bereits bestehenden Diagramm wird mitteles

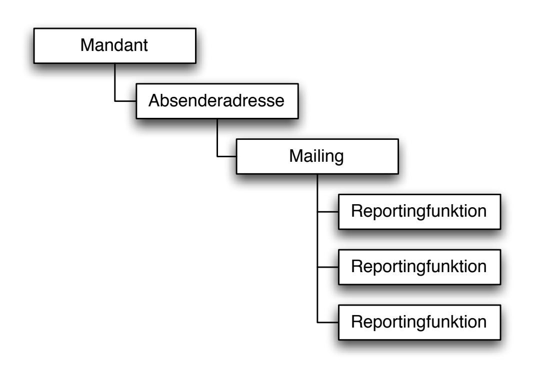


Abbildung 4.6: Baumstruktur mit Reportingfunktionen

"Drag and Drop"-Bedienung erreicht. Der prinzipielle Ablauf dieses Vorgehens wird schematisch in Abbildung 4.7 dargestellt.

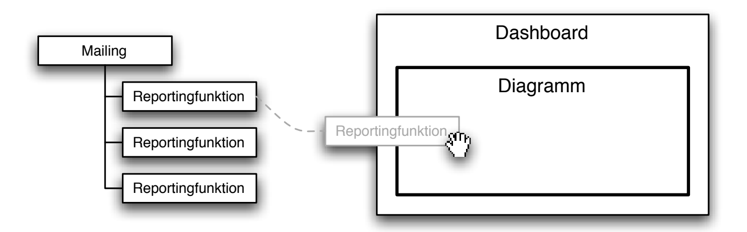


Abbildung 4.7: Baumstruktur: Drag and Drop

4.1.7 Mehrsprachigkeit

Die Abteilung NMI-TA betreut Kunden international, was die Mehrsprachigkeit erfordert. Das HTML-basierte Reporting bietet bereits mehrere Sprachen an, wobei für jeden Kunden die gewünschte Sprache in einer MSSQL-Datenbank gespeichert ist. Zur Laufzeit werden die relevanten Sprachtexte aus dieser geladen. Um ein konsistentes Erscheinungsbild zu erreichen, sollen die bereits vorhandenen Sprachdaten ebenfalls im Reportingclient verwendet werden. Hierzu bieten sich zwei Lösungsansätze an.

- **Datenbankabfrage zur Laufzeit**
  Diese Variante hat den Vorteil, dass Änderungen an den in der Datenbank abgelegten Sprachtexten sofort im Reportingclient sichtbar sind. Auf der anderen Seite wird der Traffic erhöht und die Antwortzeit verlängert sich. Grund hierfür ist die Notwendigkeit einer zusätzlichen Datenbankverbindung, um die benötigten Sprachtexte zu laden. Dieses ist einmalig für die Dauer einer Session oder bei jeder neu hinzugekommenen textuellen Beschreibung erfoderlich.

- **Erstellung lokaler Sprachdatei**
  Mit Hilfe eines PHP-Skripts wird eine Datei generiert, welche die notwendigen Sprachdaten enthält. Dieser Vorgang kann manuell oder automatisiert mit einer festgelegten Frequenz (z. B. täglich) ausgeführt werden. Dadurch erzeugt die Funktion der Mehrsprachigkeit keine weitere Netzwerklast.

Bei den Sprachtexten handelt es sich in der Regel um statische Daten, so dass deren lokale Speicherung sinnvoll ist. Der geringe Mehraufwand für die Erstellung einer Sprachdatei rechtfertigt sich durch kürzere Antwortzeiten für die Anwender.

4.2 Geo-Tracking

Eine interessante Funktion stellt das so genannte Geo-Tracking dar. Hierbei werden die geographischen Lagedaten der Empfänger in Form von Längen- und Breitengraden zurückverfolgt. Das Tracking erfolgt wie bereits in Abschnitt 2.6.2 beschrieben über einen Tracking-Server. Neben dem Datum wird zusätzlich die IP-Adresse gespeichert. Diese Angaben reichen aus, um mit Hilfe eines kostenlosen Dienstes[2] den geografischen Ort zu bestimmen. Die Genauigkeit der Lagebestimmung hängt von dem genutzten Service ab. In der Regel bieten kostenpflichtige Dienste zur jeweiligen IP-Adresse detailliertere Auskunft.
Das Geo-Tracking dient dem in Abschnitt 2.7 beschriebenen Data-Warehouse als Datenquelle. Mit Hilfe dieser Datenerhebungen lassen sich z. B. Aussagen über den Erfolg eines konkreten Mailings in einer bestimmten Region machen. Solche und weitere Aspekte fließen in Marketinganalysen ein, wodurch sich die Strategieplanung optimieren lässt.

Um die geografischen Daten zu visualisieren, wird eine digitale Weltkarte benötigt. Hierfür gibt es mehrere kostenlose Anbieter (eLettershop verwendet "Virtual Earth"[3] von Microsoft) mit unterschiedlichen Funktionen. Die benötigten Daten für das Geo-Tracking werden vom PHP-Backend als KML-Datei[4] zur Verfügung gestellt. In Flex werden diese Informationen in Form von farbigen Häufungspunkten auf der eigentlichen Karte abgebildet. Daraus ergibt sich eine Heat-Map[5] mit wenigen Farbabstufungen. Mit Hilfe dieser Art der Visualisierung wird die geographische Empfängerverteilung für den Anwender direkt ersichtlich.

[2] Der eLettershop verwendet die kostenlose Version "GeoLite City" von Maxmind (siehe `http://www.maxmind.com/app/geolitecity`)

[3] siehe `http://www.microsoft.com/virtualearth`

[4] Die "Keyhole Markup Language" (KML) besitzt die XML-Syntax und wird für die Beschreibung von geographischen Daten verwendet (siehe `http://code.google.com/intl/de-DE/apis/kml/documentation/kmlreference.html`).

[5] Eine Heat-Map bezeichnet eine grafische, ebene Darstellung von Daten, deren Werte durch Farben repräsentiert werden.

4.3 MVC-Implementierung

Im PHP-Backend dieses Projekts ist die MVC-Struktur implementiert. Weitgehend bildet der Flex-Client den View-Bereich dieses Konzeptes ab.

> "Although you can consider a Flex application as part of the View in a distributed MVC architecture, you can use Flex to implement the entire MVC architecture on the client."[6]

Hierfür eignen sich alle MVC-Frameworks, die die Programmiersprache ActionScript unterstützen. Zwei etablierte MVC Lösungen für ActionScript sind PureMVC[7] und Cairngorm[8]. Die vollständige MVC-Implementierung eignet sich vornehmlich für Projekte von komplexer Struktur. Zwar basiert der eLettershop auf einer umfangreichen Klassenarchitektur, diese ist jedoch vollständig im PHP-Backend abgebildet. Im Flex-Client sind neben den Verbindungsmethoden hauptsächlich Funktionen für die Darstellungsart zu implementieren. Erweiterungen der Reportingfunktionen sind im Wesentlichen im PHP-Backend zu realisieren. Aus diesen Gründen bleibt die Programmstruktur überschaubar. In diesem Fall wirkt sich eine MVC-Implementierung negativ aus, da sie zu einem erheblichen Mehraufwand bei der Entwicklung führt und die Komplexität erhöht. Wie eine geeignete Programmstruktur für das betrachtete Projekt, ohne Anwendung eines MVC-Frameworks, erreicht werden kann, wird im folgenden Abschnitt erläutert.

4.4 Softwarearchitektur

Programmierung besteht im engeren Sinne aus der Erstellung von Quelltext (vgl. [Pit86]). Statt der Verwendung eines restriktiven MVC-Konzeptes werden in dem betrachteten Projekt gängige Methoden der Softwareentwicklung kurz vorgestellt. Ihre konsequente Anwendung erleichtert die spätere Wartung von ActionScript-Programmcode und der MXML-Komponenten.

4.4.1 Unified Modelling Language (UML)

UML-Diagramme stellen eine wichtige Grundlage für die strukturierte Umsetzung eines Softwareprojekts dar. Auf der offiziellen Webseite[9] dieser Modellierungssprache steht:

> "The Unified Modeling Language™- UML - is OMG's[10] most-used specification, and the way the world models not only application structure,

[6] vgl. `http://livedocs.adobe.com/flex/3/html/help.html?content=introbd_2.html`, zugegriffen am 28. März 2009

[7] siehe `http://trac.puremvc.org/PureMVC_AS3`

[8] siehe `http://opensource.adobe.com/wiki/display/cairngorm/Cairngorm`

[9] vgl. `http://www.uml.org`, zugegriffen am 5. März 2009

[10] "Object Management Group" (OMG) ist ein nicht-kommerziell orientiertes Konsortium, welches u. A. IT-Spezifikationen entwickelt.

> behavior, and architecture, but also business process and data structure."

Bei UML-Elementen handelt es sich um Abstrahierungen von Funktionen und Objekten. Mit ihrer Hilfe lassen sich Programmabläufe oder z. B. Klassendiagramme kompakt und übersichtlich darstellen. Es existieren Konventionen, die die Gestalt der UML-Elemente und deren Bezeichnungen spezifizieren, um Mehrdeutigkeit auszuschließen. Dadurch werden diese Diagramme zur universellen Ausdrucksform. Dabei besitzt die UML-Darstellung eine wesentlich geringere Komplexität als das abzubildende reale System. Auf diese Weisen werden Problembereiche einfacher erkennbar.
Für das betrachtete Projekt wird in einem Aktivitätendiagramm (siehe Abb. 4.8) gezeigt, welche Aktivitäten notwendig sind, um eine Reportingfunktion zu nutzen. Der Filter kann mit individuell eingestellten oder aber mit vordefinierten Werten angewendet werden. Durch das Diagramm wird deutlich, dass, nachdem eine Reportingfunktion gewählt wurde, dem Nutzer wieder alle Instanzen des vorher durchlaufenden Workflows direkt zu Verfügung stehen. Diese Funktionalität muss umgesetzt werden, um dem Usability-Aspekt der Steuerbarkeit (vgl. Abs. 4.8.1) Rechnung zu tragen.

4.4.2 Strukturierung

Es gibt verschiedene Dateiformate, die für dieses Projekt relevant sind wie z. B. Grafiken oder eigene MXML-Komponenten. Diese Dateien gilt es in einer geeigneten Verzeichnisstruktur abzulegen, um Übersichtlichkeit zu erreichen. So erhalten z. B. die selbst erstellten MXML-Komponenten einen eigenen Ordner "components". Durch einen Namespace[11] kann den entsprechenden Komponenten ein beliebiger Präfix zugewiesen werden. In dem <mx:Application>-Tag können durch folgenden Parameter Komponenten eingebunden und das Namespacing spezifiziert werden:
`xmlns:comp="components.*"`
Für Unterordner wird die Punkt-Notation verwendet. Somit können alle Komponenten (`.*`) im Ordner "components" mit dem Tag <comp:KomponentenName> angesprochen werden.
Zusätzlich soll es eine Klasse "Const"geben, die alle benötigten Konstanten für die Anpassung der Applikation beinhaltet. So sind z. B. Datumsformate und andere Darstellungsoptionen dort zu definieren, um einen zentralen Steuerungspunkt für den Entwickler zu schaffen. Besonders hilfreich ist diese Variante für Administratoren, die die Applikation nicht näher kennen, jedoch weitreichende Modifikationen vornehmen möchten. Ihnen wird durch diese zentrale Konfigurationsmöglichkeit ein schnelles Anpassen des Systems ermöglicht. So lässt sich dort z. B. das globale Datumsformat definieren: `public static const DATE_FORMAT:String = "DD.MM.YYYY HH:SS"`.

[11] engl. für Namensraum; Namensräume werden benutzt, um Konflikte bei der Namensgebung zu vermeiden.

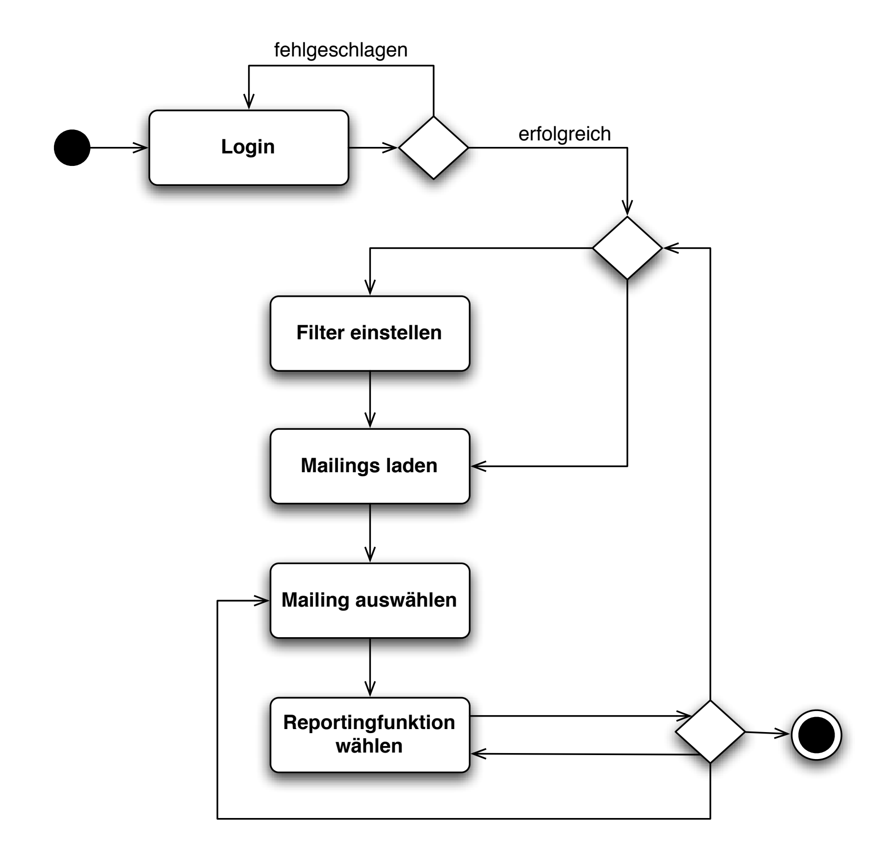


Abbildung 4.8: Aktivitätendiagramm: Mailingvergleich

4.5 Verbindungsaufbau zum PHP-Backend

Der Verbindungsaufbau von Flex zu PHP gehört zum wesentlichen Teil der Konzeption. Hier haben Sicherheit und Performance die höchste Priorität.

4.5.1 Ablauf

In Abb. 4.9 wird der Ablauf skizziert, ein ValueObject zu erhalten. Im PHP-Backend wird die Funktion einer Klasse angesprochen. In Flex bildet die Basis dafür das Remote-Objekt, in welchem der Funktionsname aus der PHP-Klasse angegeben wird. Näheres zum Remote-Objekt wird im nächsten Abschnitt erörtert.

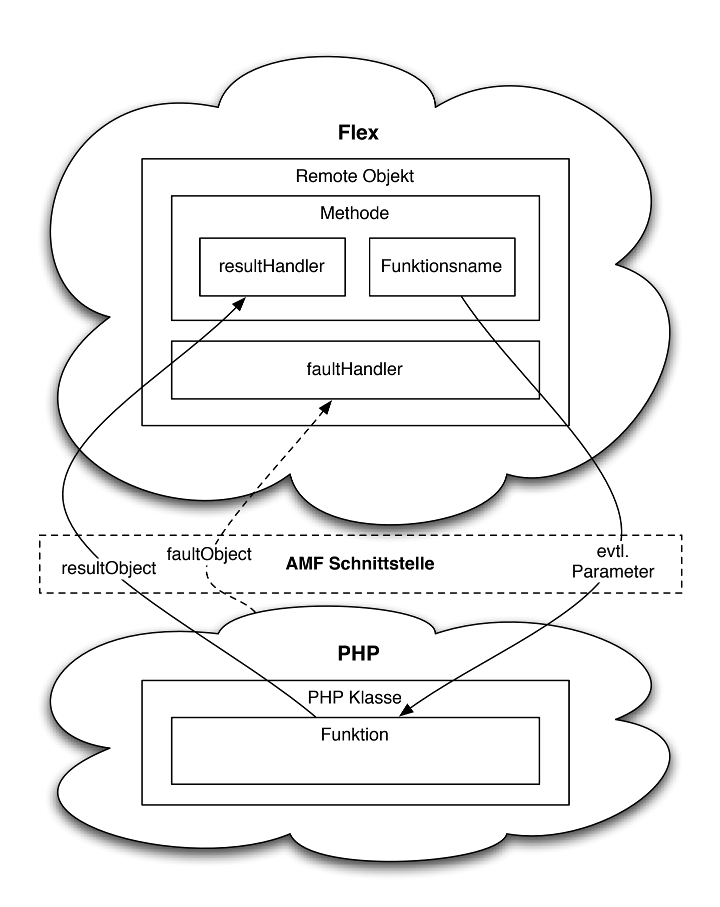


Abbildung 4.9: ValueObject erhalten

4.5.2 Remote-Objekt

Um ein Remote-Objekt erstellen und nutzen zu können, werden neben dem PHP-Backend folgende drei Bestandteile in Flex benötigt.

- **XML-Datei: services-config.xml**
  In der Datei services-config.xml wird der Endpoint[12] angegeben, der den späteren Zugriff auf die PHP-Klassen mittels Zend_Amf ermöglicht. Der Endpoint muss mindestens die folgenden Code-Zeilen enthalten, um als Schnittstelle zu fungieren.

```
<?php
  //Den Server instanziieren
  $server = new Zend_Amf_Server();
  $response = $server->handle();
  echo $response;
?>
```

Quellcode 4.1: Zend_Amf_Server

Für das Class-Mapping wird ebenfalls der Endpoint verwendet. Jede Klasse die von Flex direkt angesprochen wird, muss in folgender Form eingetragen werden.

```
//Klasse registrieren
$server->setClass("eLettershop_User_Remote");
//Class Mapping für das User Value Object
$server->setClassMap("UserVO", "eLettershop_User_Remote");
```

Quellcode 4.2: Beispiel Class Mapping UserVO

- **MXML-Objekt:** `<mx:RemoteObject>`
  In dem Remote-Objekt werden neben der aufzurufenden PHP-Funktion so genannte Handler[13] angeben, welche das Resultat entgegennehmen und verarbeiten. Man unterscheidet zwei verschiedene Arten von Handler, die nachfolgend beschrieben werden.

- **ActionScript-Funktionen:** `resultHandler(event)`, `faultHandler(event)`
  Bei erfolgreicher Verbindung und Übermittlung von Daten wird die `resultHandler(event)`-Funktion ausgeführt. Die übergebene Variable `event` ist vom Typ `ResultEvent`. Treten hingegen Übermittlungfehler auf, so wird die `faultHandler(event)`-Funktion aufgerufen. In diesem Fall werden Error-Code und Fehlermeldung in einem `FaultEvent`-Objekt zurückgegeben.

In Abb. 4.10 wird ein möglicher Ablauf für die Erstellung eines Remote-Objekts skizziert. Die eigentliche Komponente `<mx:RemoteObject>` wird im folgenden Code-Beispiel dargestellt.

[12] Beim Endpoint handelt es sich in diesem Fall um eine URL, die den Verbindungsaufbau zu PHP initialisiert.

[13] Handler sind Funktionen die aufgerufen werden, sobald ein gewisses Ereignis eintritt. In diesem Fall wird die Handler-Funktion ausgeführt, wenn das Result-Objekt vollständig empfangen wurde.

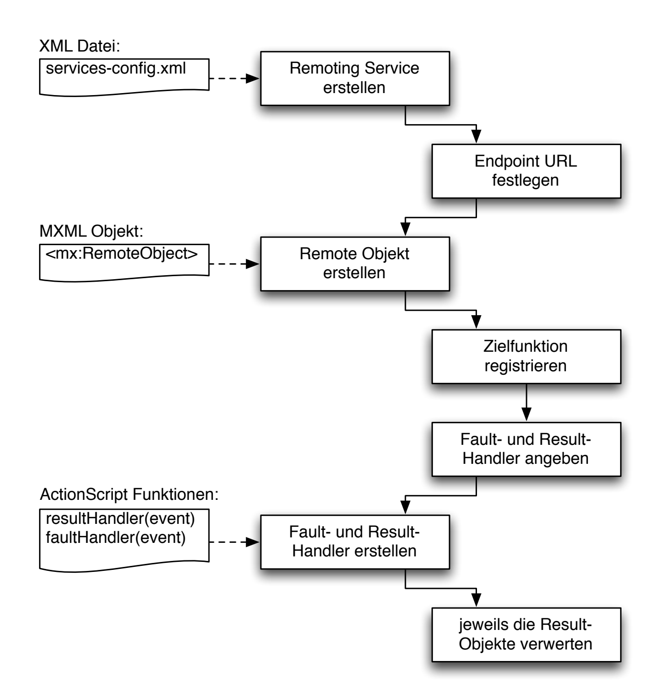


Abbildung 4.10: Remote-Objekt in Flex erstellen

```
1  <mx:RemoteObject id="amfUserRemote"
2      source="eLettershop_User_Remote"
3      destination="zend"
4      fault="faultHandler(event)">
5    <mx:method name="getUserVO" result="handleResultUser(event)"/>
6  </mx:RemoteObject>
```

Quellcode 4.3: Beispiel RemoteObject

Nach der Implementierung muss bei Nutzung einer Funktion des Remote-Objekts beachtet werden, dass das gewünschte Result-Objekt erst im Result-Handler bereitgestellt wird. Es gibt unterschiedliche Vorgehensweisen, das Result-Objekt zu verwerten. Naheliegend ist eine Auswertung direkt in dem Result-Handler (im Quellcode 4.3: `handleResultUser(event)`). Falls eine im Remote-Objekt angegebene PHP-Funktion für unterschiedliche Verwendungszwecke benötigt wird, eignet sich dieses Vorgehen nicht, da der Result-Handler wissen muss, was mit dem Rückgabewert geschehen soll. Somit muss der Rückgabewert oder ein äußerer Einfluss (z. B. globale Variable) das Verhalten bestimmen. Eine weitere Möglichkeit liegt in der Erstellung eines globalen ValueObjects, welches in dem Result-Handler gesetzt wird. Das `RemoteObject` kann innerhalb der Applikation von jeder Funktion aus über die Eigenschaft "id" (im Quellcode 4.3: `amfUserRemote.getUserVO()`) angesprochen werden. Nach Aufruf einer dem `RemoteObject` bekannten PHP-Funktion wird der Rückgabewert im Result-Handler auf das globale Value Object übertragen. Durch die asynchrone Ar-

beitsweise von Flex kann ein direkter Zugriff auf dieses Objekt im Anschluss an das Remoting kritisch sein, genau dann, wenn die Daten noch nicht vollständig übermittelt wurden. Solche vorschnellen Zugriffsversuche auf noch nicht vorhandene Rückgabewerte werden üblicherweise durch EventHandler (vgl. [Moo07]) verhindert. Diese werden einem Objekt zugewiesen und erfordern in Flex mindestens zwei Parameter. Sie beschreiben zum einen den Event (z. B. ein Klick auf die linke Maustaste), welcher abgefangen werden soll sowie zum anderen die Reaktion darauf. Allerdings besitzt dieses Konzept einige Einschränkungen bezüglich der Angabe der Rückruffunktion. Da Flex für jedes Event einen fest vordefinierten Satz an Objekten zurückgibt, ist das Senden zusätzlicher Parameter nicht möglich. Diese Restriktion erschwert den Umgang mit Events. Eine elegante Lösung der erläuterten Problematik bieten die beiden Komponenten `AsyncToken` und `ItemResponder`.

```
1  var itemResp:ItemResponder = new ItemResponder(
2    //bei erfolgreicher Übermittlung
3    function(data:Object, token:Object=null):void {
4      //Verwertung der Daten
5      var ValueObject:KlasseVO = KlasseVO(data.result);
6      //weitere Arbeiten mit dem ValueObject
7    },
8    //bei Fehler in der Übermittlung
9    function(err:Object, token:Object=null):void {
10     //Fehlerausgabe
11     Alert.show(err.toString());
12   }
13 );
14 var token:AsyncToken = remoteObject.remoteFunction();
15 token.addResponder(itemResp);
```

Quellcode 4.4: Beispiel AsyncToken und ItemResponder

Der Vorteil dieses Vorgehes liegt in der erzwungenen sequentiellen Abarbeitung. Dadurch werden die Rückgabewerte erst dann verwendet, wenn diese vollständig übermittelt sind. Hierbei ist zu beachten, dass das Remote-Objekt in der Regel ebenfalls eine Handler-Funktion für die erfolgreiche bzw. fehlgeschlagene Übermittlung enthält. Dadurch wird bei erfolgreicher Übermittlung der Rückgabewert sowohl vom `ItemResponder` als auch vom `remoteObject` verarbeitet, was sich negativ auf die Performance auswirkt. Bei fehlgeschlagener Übermittlung tritt dementsprechend eine doppelte Fehlermeldung auf. Um diese Redundanz zu vermeiden, kann im `ItemResponder` auf das Verwerten des `faultObject` verzichtet werden. Zudem muss im `remoteObject` der `resultHandler()` so angepasst werden, dass keine doppelte Auswertung des Events stattfindet.

4.6 Sicherheit

Bei den Reportinginformationen handelt es sich um sensible Kundendaten, so dass der Zugriff unauthorisierter Personen verhindert werden muss. Um eine Flex-Applikation vor ungewollten Zugriffen zu schützen, gibt es mehrere Aspekte, die

beachtet werden müssen.

4.6.1 Sicherheit ohne SSL

Im eLettershop erfolgt jegliche Datenübertragung verschlüsselt. Falls keine Möglichkeit für eine sichere Verbindung über SSL[14] besteht, können alle zwischen Client und Server ausgetauschten Daten beobachtet resp. mitgelesen werden. Mit Zend_Amf werden die Daten binär übermittelt und sind somit nicht direkt lesbar, jedoch sind die Spezifikationen für AMF frei zugänglich. Es besteht also die Gefahr, dass die übermittelten Daten mitprotokolliert und übersetzt werden.
Besonders sensible Daten sind dabei Benutzernamen und Passwörter. Falls diese abgefangen werden, ist ein Missbrauch der Kundendaten nicht auszuschließen. Eine Option, die Authentifizierung ohne SSL möglichst sicher abzuwickeln, wird im nächsten Abschnitt erläutert.

4.6.2 Authentifizierung

Ein wichtiger Aspekt beim Aufbau einer Flex-Applikation mit PHP-Backend ist die Umsetzung der Authentifizierung. Selbstverständlich sollen nur angemeldete Nutzer auf die Reportingoberfläche Zugriff haben. Zudem sollen die Kunden nur ihre eigenen Mailings einsehen können. Um dies zu gewährleisten, bedarf es einer Login-Funktion, der die potentiellen Nutzer ihre Anmeldedaten übergeben. Der Reportingclient ist sowohl direkt über eine URL als auch über das Webfrontend des eLettershops erreichbar. Dieser Umstand erfordert zwei verschiedene Authentifizierungsmöglichkeiten.

- **Login über den Reportingclient**
  Beim Start des Reportingclients werden zunächst die Anmeldedaten abgefragt. Diese Daten müssen allerdings über das Internet verschickt werden und somit tritt die in Abschnitt 4.6.1 beschriebene Problematik ein.

- **Authentifizierung mittels eLettershop-Session**
  In diesem Fall erkennt der Reportingclient, dass der Nutzer sich bereits gegenüber dem eLettershop authentifiziert hat. Somit erübrigt sich für diesen Fall die Notwendigkeit einer Login-Startseite in Flex. Diese Funktion erfordert so genannte "Sessions", welche im nächsten Abschnitt behandelt werden.

Um die erste Login-Funktion ohne Hilfe von SSL sicher gestalten zu können, eignet sich das Protokoll "Challenge-Handshake Authentication Protocol" (CHAP). Es verwendet einen 3-Wege-Handshake, dessen Ablauf in Abbildung 4.11 schematisch skizziert ist.

In Flex lässt sich diese Variante wie folgt umsetzen.

[14] SSL ist die Abkürzung für "Secure Sockets Layer" und bezeichnet ein Übertragungsprotokoll, dass die Daten verschlüsselt überträgt.

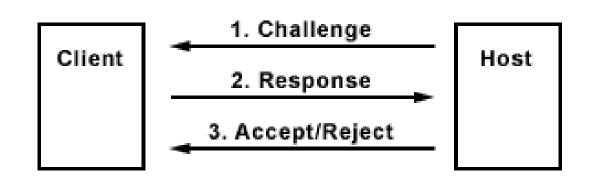


Abbildung 4.11: CHAP: 3-Wege-Handshake[15]

1. **Challenge**
   Das PHP-Backend schickt einen generierten Zufallswert (Challenge) an den Flex-Client.

2. **Response**
   Der Flex-Client bildet aus einer Kombination von Challenge und eingegebenen Benutzerdaten einen so genannten Hashwert[16]. Dieser wird an das PHP-Backend zurück geschickt.

3. **Accept/Reject**
   Parallel zu diesem Vorgang bildet das PHP-Backend auf gleiche Weise den Hashwert. Die Authentifizierung ist nur dann erfolgreich, wenn diese beiden getrennt voneinander berechneten Werte übereinstimmen.

Durch diese Methode wird eine verschlüsselte Übermittlung des Passworts erreicht. Der anschließende Datenaustausch zwischen Flex-Client und PHP-Backend ist dadurch jedoch noch nicht abgesichert. Um dies zu erreichen bietet sich u. A. ein asynchrones Verschlüsselungsverfahren[17] an.

4.6.3 Sessions

Bei HTTP handelt es sich um ein zustandsloses Protokoll. Um Zustände auf dem Server speichern zu können, wird eine so genannte Session[18] verwaltet. In dieser wird der Nutzer über eine Session-ID identifiziert. Sie erlaubt es, mehrere Zugriffe ein und desselben Anwenders zusammenzufassen. Alternativ zur serverseitigen Speicherung der Session-Daten können diese in Form von HTTP-Cookies[19] auf dem Client abgelegt werden.

[15]vgl. `http://www.elektronik-kompendium.de/sites/net/0906171.htm`, zugegriffen am 13. März 2009
[16]Ein Hashwert bezeichnet einen alphanumerischen Wert, der durch eine Hashfunktion generiert wird. Daten können mit dieser Funktion verschlüsselt werden, jedoch können die Daten ausgehend von ihren Hashwerten nicht wieder ermittelt werden.
[17]Die asynchrone Verschlüsselung erfordert zwei Schlüssel, einen privaten und einen öffentlichen. Der öffentliche Schlüssel wird benutzt, um eine Nachricht an den Besitzer dieses Schlüssels zu senden. Die verschlüsselten Daten sind daraufhin nur noch mit dem privaten Schlüssel lesbar.
[18]engl. für Sitzung
[19]HTTP-Cookies werden in der Regel bei langanhaltenden Sessions verwendet.

4.7 Darstellung

Ein wesentliches Ziel des Projektes ist die Entwicklung einer intuitiv zu bedienenden, optisch ansprechenden Benutzerschnittstelle. Zudem soll diese potentere Darstellungsmöglichkeiten bieten als die bestehende HTML-Variante. Neben Usability-Aspekten soll sich die Oberfläche an das Corporate Design[20] anlehnen. Eine vollständige Umsetzung des eLettershop-Layouts ist ungeeignet, da sich die Funktionen des Reportingclients wesentlich von denen des eLettershops unterscheiden. Beim Reportingclient handelt es sich um eine eigenständige Applikation. Das Corporate Design wird daher lediglich in Bezug auf die Farbwahl sowie die verwendeten Grafiken berücksichtigt. Hiermit wird dem Wiedererkennungswert Rechnung getragen. Dennoch sollte bei Verwendung der Farben folgendes beachtet werden:

> "Das Hauptziel der Farbdarstellung – die Erleichterung der Informationsverarbeitung – wird erreicht, wenn die gewählte Farbe leicht feststellbar, identifizierbar und unterscheidbar ist. Zudem sollte die Bedeutungszuordnung der Farbe aufgabenangemessen sein."[21]

4.7.1 Anordnung der Hauptelemente

In Absprache mit dem Auftraggeber wird ein Papier-Prototyp erstellt, welcher das generelle Layout des Reportingclients festlegt. Dieses ist in Abb. 4.12 skizziert.
Das Applikationsfenster wird in drei Bereiche aufgeteilt.

- **Menü**
  Um der Erwartungskonformität gerecht zu werden, wird das Einstellungsmenü als Leiste wie bei konventionellen Desktopanwendungen oben über die gesamte Fensterbreite platziert. Flex bietet speziell für diesen Zweck die Komponente <mx:ApplicationControlBar> an.

- **Navigation**
  Wie in den meisten Anwendungen (z. B. Web, Windows OS) werden die Elemente zur Navigation linksseitig angeordnet. Das Navigieren bezieht sich in diesem Fall jedoch nicht auf Seiten oder Dateien, sondern auf Mailings, welche in einer Baumstruktur angeordnet sind.

- **Dashboard**
  Im Dashboard[22] werden die Reportingdaten visualisiert. Als die wesentliche Funktion des Reportingclients erhält dieser Bereich den größten Oberflächenanteil. Der Individualisierbarkeit als ein Usability-Kriterium wird durch die

[20] Das Corporate Design ist ein Teilbereich der Corporate Identity (engl. für Unternehmensidentität) und bezeichnet eine einheitliche Unternehmungsdarstellung in Bezug auf ihr Erscheinungsbild.

[21] vgl. `http://www.usability.ch/News_D/farbwww.htm`, zugegriffen am 2. April 2009

[22] Die Namensgebung "Dashboard" (engl. für Armaturenbrett, Instrumententafel) wurde gewählt, da es sich um ein Feld handelt, in dem unterschiedliche Diagramme und Tabellen angezeigt und auch modifiziert werden können.

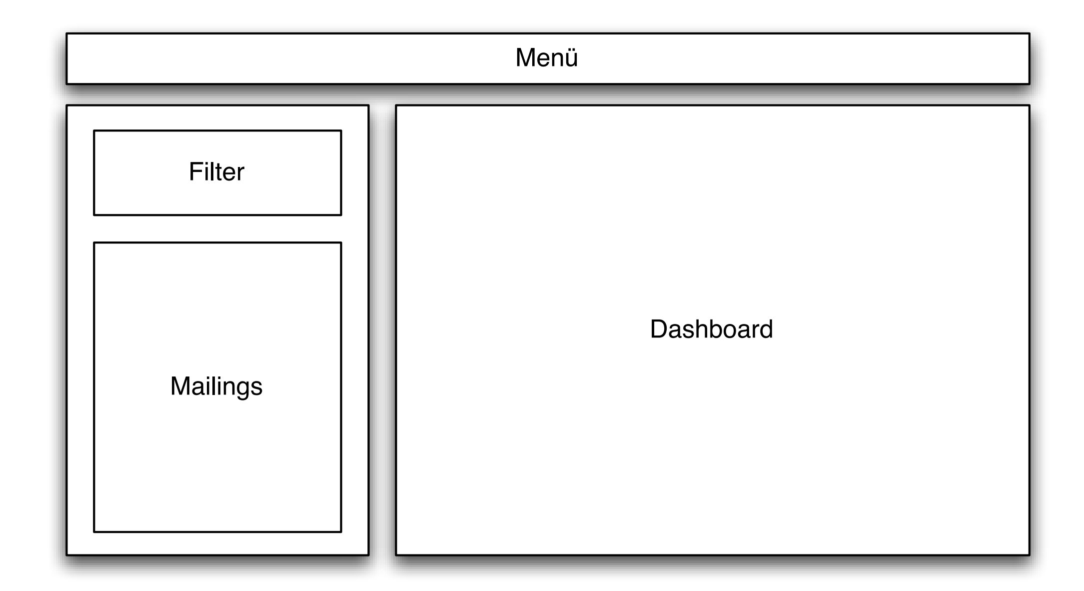


Abbildung 4.12: Papier-Prototyp

beiden folgenden Aspekte Rechnung getragen: Zum einen kann der Anwender die Größe der Darstellungselemente (z. B. Graphen) innerhalb des Dashboards frei skalieren. Des Weiteren kann die Breite der Navigationsspalte variiert werden, so dass dem Dashboard die gesamte Fensterbreite zur Verfügung steht.

Durch diese Anordnung der Hauptelemente soll eine intuitive Aufteilung gewährleistet werden.

4.7.2 Dashboard

Der größte Teil der Reportingoberfläche soll den Reportingdarstellungen zur Verfügung stehen. Die Darstellungselemente werden in Fenstern platziert, welche verschoben, in ihren Abmaßen frei skaliert sowie geschlossen werden können. Wie bereits in Abschnitt 4.1.6 erläutert wurde, werden darzustellende Datensätze per Drag and Drop dem entsprechenden Darstellungselement hinzugefügt. Die direkte Bearbeitung von Graphen (z. B. das Entfernen eines Datensatzes) geschieht mit Hilfe von Steuerelementen. Diese werden zusammen mit den Darstellungselementen in grafischen Modulen (Fenstern) gekapselt, so dass die Zuordnung offensichtlich und die Bedienung entsprechend intuitiv ist.
Der Anwender kann mehrere Reportingfunktionen innerhalb des Dashboards betrachten und dadurch vergleichen. Für die Umsetzung des Feldes eignet sich die Komponente <mx:Tile>, welche Darstellungsobjekte gekachelt anordnet. Für die Größe der einzelnen Felder, resp. Diagramme, ergibt sich ein Layout-Problem. Jedes Feld in der Komponente Tile besitzt die gleichen Maße, die durch die Parameter

`tileHeight` und `tileWidth` angegeben werden. Diese müssen zur Laufzeit so angepasst werden, dass stets alle Diagramme sichtbar sind und der Platz im Dashboard effizient genutzt wird. Scrollbalken sollten vermieden werden, da diese zusätzliche Aktivität des Anwenders erfordern.
Laut Kundenbefragung werden maximal vier Diagramme gleichzeitig betrachtet. Dieser Wert sollte jedoch keine feste Vorgabe darstellen, da sich die Kundenanforderungen ändern können. Die Implementierung erlaubt daher die gleichzeitige Anzeige von bis zu acht Feldern. Eine höhere Anzahl erscheint nicht sinnvoll, da auf handelsüblichen Monitoren aus Platzgründen keine nutzbare Darstellung erreicht werden kann. In Abb. 4.13 werden das Tile-Layout skizziert und die relevanten Eigenschaf-

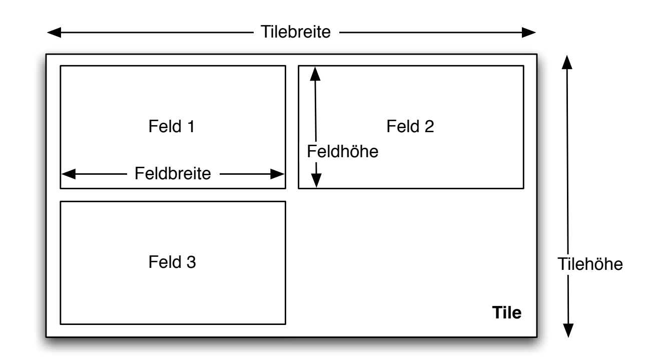


Abbildung 4.13: Layout der Tile-Komponente

ten dargestellt. Es muss anhand der Diagramm-Anzahl die universelle Feldgröße stets neu berechnet werden. Da der Trend bei Bildschirmen in Richtung Widescreen geht, bietet sich eine Anordnung an, welche die Felder nebeneinander auflistet. Ab drei Feldern soll eine zweite Zeile für die Diagramme hinzukommen. Die prinzipielle Lösung dieser Aufgabe kann durch folgenden Pseudo-Code beschrieben werden:

```
1  // Die Höhe des Feldes bestimmen
2  wenn (AnzahlDiagramme > 2) dann
3    Feldhöhe = Tilehöhe / 2 // zwei Zeilen
4  sonst
5    Feldhöhe = Tilehöhe // eine Zeile
6  ende wenn
7
8  // Die Breite des Feldes bestimmen
9  wenn (AnzahlDiagramme > 4) dann
10   // Spaltenaufteilung in zwei Zeilen
11   // bei gerader Anzahl: oben und unten gleiche Anzahl
12   // bei ungerader Anzahl: oben ein Diagramm mehr als unten
13   Feldbreite = Tilebreite / Abrunden((AnzahlDiagramme + 1) / 2))
14 sonst
15   Feldbreite = Tilebreite / 2 // zwei Spalten
16 ende wenn
```

Quellcode 4.5: Tile Layout

Der endgültige in ActionScript Code programmierte Algorithmus berücksichtigt einen zusätzlichen Aspekt. Um einen Abstand zwischen den Feldern zu erreichen, werden einige Pixel von der Tilehöhe und -breite substrahiert (vgl. Abb. 4.13).

4.7.3 Barrierearme Flash-Applikationen

Im Internet können unterschiedliche Barrieren auftreten. Neben technischen Hindernissen wie z. B. Plugins, existieren auch personenspezifische Barrieren. So kann eine Sehschwäche dazu führen, dass eine Webseite mit zu kleinen Navigationselementen nicht bedienbar ist. Dabei liegt die Verantwortung beim Entwickler, die Webseiten so zu gestalten, dass sie für möglichst viele Menschen zugänglich sind (vgl. [Hel05]). Tim Berners-Lee betont die Relevanz dieses Aspektes auf der Webseite für die "Web Accessibility Initiative" (WAI) des W3-Konsortiums[23] mit folgender Aussage:

> "The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect."[24]

Laut einer Untersuchung zum deutschsprachigen Web von Validome[25], einem Validierungs-Service für HTML, sind nur 3,9% der betrachteten Webseiten W3C-konform (vgl. Abb. 4.14). Die restlichen 96,1% weichen von dem W3C-Standard ab,

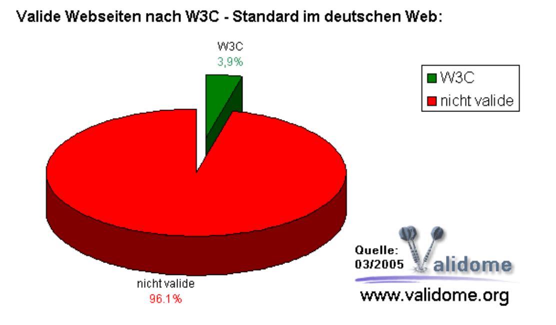


Abbildung 4.14: Überprüfung auf W3C-Standardkonformität

so dass es potentiell zu Problemen bei der Verwendung von Hilfsprogrammen wie z. B. einem Screen-Reader[26], kommen kann.
Auch die deutsche Initiative Aktion Mensch befasst sich mit diesem Thema. In einem Artikel über die Verwendung von Flash stellt sie folgendes fest:

> "Wer hätte gedacht, dass man mittlerweile Flash zusammen in einem Satz mit Barrierefreiheit erwähnen darf? Das Programm hat sich von einer verpönten Spielwiese zu einem ernstzunehmenden Werkzeug für

[23] Das "World Wide Web Consortium" (W3C) entwickelt Standards, Richtlinien und Technologien für das WWW (siehe `http://www.w3.org`).
[24] vgl. `http://www.w3.org/WAI`, zugegriffen am 12. April 2009
[25] vgl. `http://www.validome.org`, zugegriffen am 12. April 2009
[26] Ein Screen-Reader ist in diesem Fall ein Programm, das Inhalte einer Webseite akustisch wiedergeben kann.

> die Erstellung zugänglicher Rich-Media-Inhalte gemausert. Der englische Accessibility-Experte Mike Davies meint dazu: 'Its time to admit, Flash is part of the web accessibility toolbox.' "[27]

Flash-Applikationen können durch ihre audiovisuellen Möglichkeiten auch sehbehinderten Menschen eine barrierearme Anwendung bieten. Dies lässt sich z. B. umsetzen, indem gängige Hilfsprogramme durch Flash-Komponenten (z. B. eingebettete Screen-Reader oder Tastatursteuerung) ersetzt werden. Bei der technischen Umsetzung ist besonders auf eine intuitive Bedienung zu achten. Im günstigsten Fall werden Rechner, welche mit speziellen Hilfsprogrammen ausgestattet sind, obsolet. Der Anwender erlangt dadurch eine größere Unabhängigkeit bezüglich des verwendeten Rechners sowie des eingesetzten Betriebssystems.
Unter der Prämisse, dass kein Anwender vom Zugang zu einer Webseite ausgeschlossen werden soll, sind die Begriffe Barrierefreiheit und Usability untrennbar miteinander verbunden. Wie sich benutzerfreundliche Anwendungen entwickeln lassen, wird im nächsten Abschnitt erläutert.

4.8 Usability Engineering

Szwillus definiert in [Szw08] den Begriff "Usability Engineering" als "systematisches, ingenieurmäßiges Entwickeln gut benutzbarer bzw. gebrauchstauglicher Benutzungsschnittstellen".
Die von Nielsen geäußerte scharfe Kritik bezüglich der Usability von Flash-Applikationen (vgl. Abs. 3.1.2) soll im Folgenden entkräftet werden. Hierzu werden wesentliche Bereiche des Usability Engineerings angesprochen. Dabei wird speziell auf den Reportingclient sowie in allgemeiner Form auf Flash-Applikationen Bezug genommen.

4.8.1 Ergonomische Gestaltung

Im zehnten Teil der Europäischen Norm "Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten" (EN ISO 9241-10)[28] werden die Grundsätze der ergonomischen Dialoggestaltung definiert. Diese werden im Folgenden näher erläutert. Weiterführende Beispiele findet der geneigte Leser in [Szw08].

- **Aufgabenangemessenheit**
  Es sollten nur Objekte angezeigt werden, die für den Benutzer relevant sind. Daher sind z. B. nicht verfügbare Reportingfunktionen einzelner Mailings auszublenden.

[27] vgl. `http://www.einfach-fuer-alle.de/podcast/2006/kw51`, zugegriffen am 12. April 2009
[28] vgl. `http://www.interactive-quality.de/site/DE/int/pdf/ISO_9241-10.pdf`, zugegriffen am 13. April 2009

- **Selbstbeschreibungsfähigkeit**
  Wesentlich für die intuitive Benutzung ist die unmittelbare Verständlichkeit aller Bedienelemente. Die Benutzung der Applikation sollte ohne Handbuch oder andere externe Informationsquellen möglich sein. Hilfe-Texte oder Formatvorgaben sind zwei Mittel, die helfen können, Missverständnissen vorzubeugen. Außerdem sind nach Benutzerhandlungen, sofern zweckmäßig, Rückmeldungen zu geben.

- **Steuerbarkeit**
  Dem Nutzer soll ein individueller Prozessablauf ermöglicht werden. Dazu gehört z. B. die Wiederaufnahme von abgebrochenen Dialogen. Zudem sollte er Ansichten anpassen und die Ablaufgeschwindigkeit selbst bestimmen können. Im Reportingclient wird z. B. ein einstellbarer Filter existieren, der nur die gewünschten Mailings anzeigt, wodurch eine kompakte und somit übersichtliche Ansicht erreicht wird.

- **Erwartungskonformität**
  Eine erwartungskonforme Applikation zeichnet sich dadurch aus, dass die Arbeitsabläufe in einer dem Nutzer bekannten Form gestaltet werden. Die Benutzer des Reportings kennen bereits das eLettershop-Frontend. Dort verwendete Begriffe sollten daher im Reportingclient wieder aufgegriffen werden. Die Darstellungsart wird nicht dem eLettershop-Frontend gleichen, da es nur eingeschränkte Möglichkeiten bietet. Dennoch sollen die üblichen Erfahrungen mit Webanwendungen und Desktopsoftware ausreichen, um die Applikation intuitiv nutzen zu können.

- **Fehlertoleranz**
  Grundsätzlich sollten fehlerhafte Benutzereingaben abgefangen werden. Hierbei kann es sich um Daten (z. B. ungültige Datumsformate) wie auch unvorhergesehene Benutzeraktionen handeln. Eine Maßnahme ist die gezielte Benutzerführung. In der Flex-Applikation werden neben einer Datumsauswahl, die über einen Kalender erfolgt, Comboboxen zur Visualisierung und Auswahl anwendbarer Funktionen eingesetzt. Falls dennoch Fehler auftreten, sind aussagekräftige Fehlermeldungen wichtig. Das Benutzerverhalten darf unter keinen Umständen die Stabilität des Systems beeinflussen.

- **Individualisierbarkeit**
  Individuelle Anpassungen der Applikation sollten unterstützt werden. Im Reportingclient sind relevante Anzeigeelemente frei skalierbar bis hin zur Vollbildanzeige. Um die Barrierefreiheit zu berücksichtigen, ist eine optionale Schriftvergrößerung sinnvoll.

- **Lernförderlichkeit**
  Die Applikation sollte so einfach wie möglich aufgebaut sein, um schnell erlernbar zu sein. Schwer verständliche Komponenten werden durch Hilfetexte beschrieben, die den Lernprozess des Nutzers unterstützen.

Diese Grundsätze können als Leitlinien für die Erstellung und Bewertung von Benutzeroberflächen eingesetzt werden. Dabei sollte folgende Aussage beachtet werden.

> “Die Art und Weise, in der jeder einzelne Grundsatz der Dialoggestaltung umgesetzt werden kann, hängt von den Merkmalen das Benutzers, für den das Dialogsystem gedacht ist, den Arbeitsaufgaben, der Arbeitsumgebung und der jeweils eingesetzten Dialogtechnik ab.”[29]

In diesem Fall kann in Betracht gezogen werden, dass die Bestandskunden den eLettershop und seine Benutzerführung bereits kennen.

4.8.2 Heuristiken

Jakob Nielsen hat zehn Heuristiken aufgestellt, welche die Gebrauchstauglichkeit einer Software sicherstellen sollen (vgl. [Nie94], Kapitel 5). In der folgenden Aufzählung wird jede einzelne von ihnen kurz erläutert (vgl. [Szw08]) und auf den Reportingclient bezogen.

1. **Simple and Natural Dialogue**
   Ein einfacher und natürlicher Dialog zeichnet sich u. A. durch Übersichtlichkeit aus. In der Benutzeroberfläche des Reportings werden dementsprechend möglichst wenig Elemente zur gleichen Zeit sichtbar sein. Zudem sollen nicht relevante Bereiche vom Anwender ausgeblendet werden können.

2. **Speak the Users’ Language**
   Da die Abteilung NMI-TA auch ausländische Kunden bedient, wird die Applikation mehrsprachig aufgebaut. Somit können Nutzer die Texte in ihrer bevorzugten Sprache lesen. Nielsen spricht allerdings nicht nur von der Sprache an sich, sondern z. B. auch von der Vermeidung unbekannter Abkürzungen.

3. **Minimize User Memory Load**
   Um die Gedächtnisbelastung zu minimieren, ist es u. A. empfehlenswert, bei Datumseingaben das Format vorzugeben. Der Nutzer sollte durch das Programm geführt werden. So erfolgt im Falle des Reportingclients die Datumseingabe über einen integrierten Kalender.

4. **Consistency**
   Konsistenz kann z. B. durch statische Elemente, die stets an der gleichen Stelle vorzufinden sind, erreicht werden. In der Flex-Applikation wird sich das Auswahlmenü wie im eLettershop stets links befinden. Trotz Skalierbarkeit einzelner Elemente bleibt die Struktur des Layouts gleich.

5. **Feedback**
   Der Nutzer soll stets über aktuelle Ereignisse informiert werden. Bei längeren

[29] vgl. `http://www.interactive-quality.de/site/DE/int/pdf/ISO_9241-10.pdf`, zugegriffen am 12. April 2009

Ladezeiten oder aufgetretenen Fehlern muss der Nutzer klare Rückmeldungen vom System erhalten.

6. **Clearly Marked Exits**
   Um dieser Heuristik gerecht zu werden, sollten alle Dialog-Fenster und die Applikation an sich einen klar erkennbaren Ausstiegspunkt (z. B. "Abbrechen"-Button) anbieten. Aktionen, die mehr als zehn Sekunden in Anspruch nehmen, müssen abbrechbar sein. Durch das asynchrone Arbeiten einer Flex-Applikation können parallel zu Ladevorgängen weitere Aktionen ausgeführt werden. Falls z. B. der Datenbank-Server überlastet ist und die Übermittlung der Klickdaten viel Zeit in Anspruch nimmt, kann der Nutzer den Reportingclient weiterhin nutzen.

7. **Shortcuts**
   Für versierte Nutzer einer komplexen Applikation ist es sinnvoll "Shortcuts" anzubieten, um einen schnelleren Zugriff zu ermöglichen. In dieser Flex-Applikation ist diese Heuristik nicht von großer Bedeutung, da nur wenige Zugriffspunkte existieren. Dennoch werden Tastenkombinationen für einige Aktionen zur Verfügung gestellt (z. B. für das Ausrichten der Fenster).

8. **Good Error Messages**
   Falls ein Anwendungs- oder Systemfehler auftritt, sollte der Nutzer das Problem nachvollziehen können. Hierfür bedarf es der Implementierung eindeutiger und aussagekräftiger Fehlermeldungen.

9. **Prevent Errors**
   Der Dialog mit dem Anwender ist so zu gestalten, dass Anwendungsfehlern vorgebeugt wird. Durch praxisnahe Usability-Tests lassen sich ungeeignete Dialoge ermitteln. Diese Informationen fließen in den Entwicklungsprozess ein. Zusätzlich kann aus Erfahrungen im Umgang mit dem bereits existierenden HTML-Reportingclient zurückgegriffen werden.

10. **Help and Documentation**
    Mehrdeutige Abläufe oder schwer verständliche Bedienelemente können mit Tool-Tips[30] versehen werden. Eine weitere Möglichkeit sind statische Hilfe-Texte, die den Dialog erklären. Hierbei ist jedoch auf Übersichtlichkeit zu achten. Im Reportingclient soll dem Anwender möglichst viel Darstellungsfläche zur Verfügung stehen, weshalb sich Tool-Tips in diesem Fall besser eignen.

Die genannten Heuristiken sollen in der Konzeptionsphase dabei helfen, die Applikation benutzerfreundlich zu gestalten.

[30] Ein Tool-Tip ist in der Regel ein Textfeld, welches beim Verweilen der Maus über einem Objekt erscheint und darüber informiert.

4.8.3 Usability in Rich Internet Applications

Im Gegensatz zu einer herkömmlichen HTML-Webseite ergeben sich durch die Verwendung einer RIA zusätzliche Interaktionsmöglichkeiten. Ob ihre Nutzung einen positiven Einfluss auf die Usability hat, hängt u. A. von der Umsetzung ab. Es handelt sich meist um neuartige Bedienelemente, die dem Anwender nicht vertraut sind, was sich negativ auf die Erwartungskonformität auswirken kann. Um diesen Effekt zu kompensieren, ist bei der Konzeption der Benutzerschnittstelle verstärkt auf die Selbstbeschreibungsfähigkeit zu achten. Falls keine intuitive Bedienung durch den Anwender möglich ist, muss dieser entsprechend instruiert werden.
Im nächsten Abschnitt soll daher ein Usability-Test durchgeführt werden, um die Benutzerfreundlichkeit der Reporting-Applikation zu überprüfen, und ggf. Änderungen zu implementieren.
Eine Orientierungshilfe für das Usability-Engineering in RIAs bietet der Nielsen Norman Group Report "Usability of Rich Internet Applications and Web-Based Tools: Design Guidelines Based on User Testing of 46 Flash Tools"[31].

4.9 Empirische Evaluierung

Um die Gebrauchstauglichkeit einer Applikation empirisch zu testen, ist es sinnvoll, Personen mit verschiedenen Vorkenntnissen zu involvieren. Eine Liste von Heuristiken wurde in Abschnitt 4.8.2 bereits erarbeitet. Für eine professionelle heuristische Evaluation, die Usability Experten involviert, fehlen jedoch die finanziellen Mittel. Dennoch lässt sich im kleinen Rahmen eine empirische Evaluation durchführen. Voraussetzung dafür ist eine zumindest teilweise lauffähige Applikation.

4.9.1 Prototyp

Ein Software-Prototyp wird erstellt, um bereits in frühen Entwicklungsphasen Feedback bezüglich der Umsetzung zu erhalten.

> "Frühe Fehler sind teuer!"[32]

Deshalb sollten strukturelle Fehler in der Benutzerführung möglichst frühzeitig erkannt werden. Hierfür kommt das horizontale Prototyping[33] zum Einsatz. Aus diesem Grund wird eine Version des Reportingclients entwickelt, welche lediglich die Programmoberfläche und grundlegende Dialogabläufe abbildet. Der Flex-Builder ermöglich eine solche Umsetzung mit geringem Zeitaufwand. Die Verbindung zum PHP-Backend ist in diesem Fall noch nicht relevant, da der Reportingclient zunächst

[31] siehe `http://www.nngroup.com/reports/flash`
[32] vgl. [Szw08]
[33] Horizontales Prototyping wird in der frühen Design-Phase angewandt. Es dient dem Test der grafischen Benutzerschnittstelle, der Navigation im System sowie der Anordnung der einzelnen Funktionen.

mit Dummy-Daten[34] arbeiten kann. Dabei ist zu beachten, dass die Testumgebung zunächst keine nennenswerten Latenzzeiten verursacht. Um den Usabilty-Test realistisch zu gestalten, sind entsprechende Totzeiten zu simulieren, wie sie durch die Datenverarbeitung des PHP-Backends zu erwarten sind. Dieses Verhalten bietet dem Probanden die Möglichkeit des asynchronen Arbeitens und dem Entwickler die Untersuchung zweier Aspekte.

- **Plausibilitätsprüfung**
  Es soll untersucht werden, ob Probleme entstehen, falls mehrere Prozesse in kurzer zeitlicher Abfolge initiiert und parallel ausgeführt werden. Neben der Software-Technik, die solche Aufgaben leisten muss, ist das Verständnis des Anwenders diesbezüglich zu untersuchen.
- **Erwartungskonformität**
  Entscheidend für die weitere Entwicklung ist, dass der Nutzer die Funktionsmöglichkeiten erkennt. Auf die Option, bei Totzeiten weitere Funktionen ausführen zu können, muss evtl. hingewiesen werden.

Bei der Erstellung des Prototypen werden die bereits genannten Grundsätze ergonomischer Gestaltung berücksichtigt.

4.9.2 Vorbereitung und Ablauf

Der Usability-Test soll Aufschluss über mögliche Usability-Probleme geben. Im Folgenden werden nötige Vorbereitungen und Ablaufkriterien aufgezählt.

1. **Probandenauswahl**
   Wie bereits erwähnt, sollen verschiedene Personengruppen an dem Usability-Test teilnehmen. Da der Test nur abteilungsintern durchgeführt werden kann, ist die Teilnehmeranzahl stark begrenzt.
2. **Vorkehrungen**
   Laut Szwillus ist das oberste Ziel der Vorbereitung das “Schaffen absolut gleicher Vorbedingungen für alle Testpersonen!”. Aufgabenstellung und Ausgangssituation sind daher für alle Probanden identisch. Der Test sollte nicht in einer Prüfungsatmosphäre erfolgen. Um das zu erreichen wird der Hinweis “Fehler, die Sie machen, sind Schwächen des Produkts, nicht Ihre!” von Tognazzini[35] (vgl. [Szw08]) umgesetzt.
3. **Aufgabenstellung**
   Der Inhalt der Aufgabe beschränkt sich auf den Vergleich zweier Mailings. Hierdurch werden die wesentlichen Funktionen des Reportingclients angewendet und gleichzeitig getestet.

[34] Dummy-Daten sind in der Regel willkürlich erstellte Daten, die zu Testzwecken verwendet werden.

[35] Bruce Tognazzini ist ein Berater im Bereich Usability und arbeitet in der Nielsen Norman Group, die sich auf das Thema Mensch-Computer-Interaktion spezialisiert hat.

4. **Beobachtung**
   Die Probanden werden gebeten, die "Think-Aloud"-Technik anzuwenden. Das bedeutet, dass die Gedanken des Nutzers in jedem Schritt der Aufgabenstellung laut geäußert und protokolliert werden. Während des Tests ist die direkte Kommunikation zwischen Proband und Beobachter zu vermeiden, um den Testverlauf nicht zu verfälschen. Die Rückmeldungen ermöglichen den Abgleich zwischen der realisierten Benutzerführung und dem tatsächlich eintretenden Benutzerverhalten. Auftretende Diskrepanzen erlauben Rückschlüsse bezüglich erreichter Selbstbeschreibungsfähigkeit sowie Erwartungskonformität.

5. **Feedback**
   Nachdem die Aufgabenstellung abgearbeitet wurde, findet ein kurzes Gespräch mit dem Probanden statt. Dort hat dieser die Möglichkeit, Unklarheiten oder Schwierigkeiten bei der Abarbeitung sowie Kritik (z. B. Änderungswünsche) zu äußern.

4.9.3 Aufgabenstellung

Jeder Proband erhält die gleiche Begrüßung und das gleiche Suchproblem. Folgende Einführung wird dem Probanden vorgetragen.

Hallo!
Diese Untersuchung findet im Rahmen meiner Diplomarbeit statt und dient der Aufdeckung von Usability-Problemen. Konkret geht es um den Test einer Reporting-Applikation für den E-Mail-Massenversand, die das Empfängerverhalten in Form von Graphen visualisiert. Es soll z. B. dargestellt werden wie viele Empfänger die E-Mail geöffnet haben.
Mit Deiner Hilfe möchte ich feststellen, ob das Programm ohne Vorkenntnisse und Hilfestellungen benutzbar ist. Nicht du wirst getestet, sondern das Programm. Du kannst also ganz entspannt die Aufgabenstellung angehen. Auftretende Probleme helfen mir, die Applikation zu verbessern. Es wäre hilfreich, wenn du deine Gedanken während des Arbeitens aussprichst. Also ruhig erwähnen, was du gerade tust, wie du es erreichen willst und was dabei unklar ist.

Der Ablauf wird folgendermaßen verlaufen:
Ich bitte Dich zu versuchen, gleich eine kleine Aufgabe zu bearbeiten. Ich werde während der Bearbeitung anwesend sein, jedoch weder eingreifen noch Fragen beantworten, um das Ergebnis nicht zu verfälschen. Anschließend findet eine kurzes Gespräch statt, in dem du mir deine Erfahrungen und evtl. Verbesserungsvorschläge mitteilen kannst.
Vorab schon mal vielen Dank für die Teilnahme!

Abbildung 4.15: Mündlicher Vortrag

Anschließend soll folgende Aufgabenstellung Usability-Probleme in der Reporting-Applikation aufdecken.

Vor dir siehst du den Prototypen der Reporting-Applikation. Ab jetzt kannst du bereits laut denken, also mir mitteilen, was du siehst und was du davon hältst.
Stell Dir vor, der Besitzer einer Webseite zu sein, die Surfbretter verkauft. Am 17.03.09 hast Du ein Mailing mit dem Betreff "Neue Surfbretter" verschickt. Eine Woche davor (10.03.09) hat die gleiche Empfängerliste von dir eine E-Mail mit dem Betreff "Günstige Surfbretter" erhalten. Nun möchtest du die Reaktionen der Empfänger gerne vergleichen:

1. **Response-Übersicht**
   Lass dir von beiden Mailings jeweils die Response-Übersicht anzeigen

2. **Klicks pro Stunde**
   Versuche nun die Klicks pro Stunde beider Mailings in einem Diagramm zu vergleichen

Abbildung 4.16: Aufgabenstellung

MailingId	Sendedatum	Absender	Titel
5221	10.03.09	info@surf.de	Neue Surfbretter
5234	17.03.09	info@surf.de	Günstige Surfbretter

Abbildung 4.17: Mailings

4.9.4 Fazit und Analyse

Acht Personen nahmen am Usability-Test teil, von denen fünf mit dem eLettershop und dem ursprünglichen HTML-Reporting vertraut sind (Gruppe 1). Bei den restlichen drei Teilnehmern (Gruppe 2) handelt es sich um zwei Praktikanten und eine Sekretärin. Das Alter der Probanden liegt zwischen 19 und 37 Jahren.
Zunächst soll in einer Auflistung auf generelle Probleme, die bei der Abarbeitung der Aufgaben protokolliert wurden, eingegangen werden.

- **Aufgabenstellung**
  Die Probanden der Gruppe 2 waren mit dem Begriff "Response-Übersicht" nicht vertraut. Dies hat zu leichten Verwirrungen innerhalb dieser Gruppe geführt.

- **Filtereinstellung**
  Um die entsprechenden Mailings auszuwählen, musste die Datumseingrenzung angepasst werden. Ein Praktikant hat dies zunächst nicht bedacht und konnte somit anfangs die Mailings nicht finden. Generell waren für Gruppe 2 die Filtereinstellungen verwirrend.

- **Mailingauswahl**
  Die Mailingauswahl konnte im Allgemeinen von beiden Gruppen schnell durchgeführt werden. Die meisten versuchten, die einzelnen Knoten des Mailing-Baums durch Doppelklick auf die Bezeichnung und nicht über die Pfeile zu öffnen. Da zu diesem Zeitpunkt noch keine Doppelklick-Funktion implementiert war, führte dies zu leichten Verzögerungen.

- **Response-Übersicht anzeigen**
  Diese Funktion haben alle Beteiligten ohne längere Überlegungen ausführen können. Den Probanden war das parallele Arbeiten während der Datenanforderung vom Server nicht vertraut, so dass sie die Totzeit nicht für das Ausführen weitere Funktionen genutzt haben.

- **Klicks pro Stunde in einem Diagramm**
  Diese Aufgabe konnten nur drei Probanden (alle aus Gruppe 1) erfüllen. Andere versuchten beide Diagramme separat anzuzeigen um anschließend eines der Fenster dem anderen durch Verschieben hinzuzufügen. Die übrigen Probanden resignierten relativ schnell bei dieser Aufgabe. Die Möglichkeit des Drag and Drop der Reportingfunktionen wurde somit weitgehend nicht erkannt.

Bei der Auswertung ist zu beachten, dass es sich um eine geringe Anzahl von Testpersonen handelte und der eLettershop einigen Probanden noch fremd war. Die Zielgruppe ist jedoch mit dem eLettershop vertraut und besitzt somit ein Grundverständnis für Mailings und deren Reportingfunktionen. Dennoch konnte ein klares Defizit bei der intuitiven Bedienung des Mailingvergleichs festgestellt werden, da die Benutzer die Existenz der Drag and Drop Funktion nicht erwartet bzw. falsch angewendet haben.
Die Komplikationen mit unbekannten Begriffen (z. B. Response-Übersicht) und den Filtereinstellungen traten lediglich in der Gruppe 2 auf. Da die Nutzer des Reportingclients mit den Begriffen vertraut sind und auch die Filterfunktion kennen, ist diese Problematik für die Bestandskunden nicht weiter relevant. Neukunden steht ein entsprechendes Handbuch für den eLettershop zur Verfügung, welches grundlegende Funktionen und Begriffe erklärt. Im Gegensatz dazu bedarf es im Falle der im Reportingclient vorgesehenen Drag and Drop Funktion einer Einweisung in Form von Hilfetexten oder eines Einführungsvideos. Da Flex eine Komponente <mx:VideoDisplay> für die Wiedergabe von Videos bereitstellt, bietet es sich an, eine kurze visuelle Demonstration zum Funktionsumfang des Reportingclients einzubetten. Für die Erzeugung des Videos eignet sich ein Screen-Capture Programm, welches die Programmoberfläche mit den Bewegungen der Maus aufzeichnet. Der

Transferaufwand für den Benutzer wird dadurch gering gehalten, indem das Demonstrationsvideo mit Hilfe der tatsächlichen Oberfläche des Reportingclients erstellt wird. Um das Videomaterial mit weiteren Informationen anzureichern, können diesem Texte und Sprache hinzugefügt werden. Um den Benutzer nicht zu entmündigen, wird das Video nur aufgrund einer Benutzerinteraktion angezeigt und lässt sich zudem jederzeit abbrechen.
Die Probanden nutzten im anschließenden Gespräch die Möglichkeit, neben den laut geäußerten Gedanken weiteres Feedback zu geben.

- **Positiv**
  Besonders positiv fiel die Baumstruktur auf, welche die Navigation wesentlich übersichtlicher gestaltet als im eigentlichen eLettershop. Zudem wurde die Visualisierung der Graphen und die generell flüssige Darstellung gelobt. Das Design des Prototyps entspricht nicht der endgültigen Optik, jedoch stieß diese bereits auf hohe Akzeptanz.

- **Negativ**
  Abgesehen von der Filternutzung und der Drag and Drop Funktion wurden keine negativen Kritikpunkte geäußert.

- **Optimierungen**
  Die Wünsche einiger Probanden wurden aufgegriffen, um den Reportingclient weiter zu verbessern. Zum Einen wurde vorgeschlagen, dass die Knoten in der Baumstruktur auch durch Doppelklick geöffnet werden können und jede Hierarchieebene zudem durch ein spezielles Bild repräsentiert wird. Zum Anderen wurde angeregt, Fenster im Dashboard frei positionieren zu können. In dem Prototypen wird die Komponente Tile verwendet, welche die Fenster automatisch kachelt.

Im nachfolgenden Kapitel werden die von den Probanden geäußerten Kritikpunkte und Anregungen berücksichtigt und in adäquater Weise technisch umgesetzt.

5 Realisierung

Das vorliegende Kapitel gliedert sich in drei Abschnitte. Zunächst werden einige verwendete Programmierkonzepte erläutert. Daran anschließend erfolgt die Präsentation der Oberfläche anhand von Screenshots. In diesem Zusammenhang werden die implementierten Bedienkonzepte vorgestellt und deren Einfluß auf die Usability bewertet. Abschließend werden auf die wenigen aufgetretenen Pitfalls hingewiesen und alternative Wege aufgezeigt.

5.1 Programmierung

In diesem Abschnitt sollen einige grundlegende Programmierkonzepte in Flex aufgelistet werden, die bei der Realisierung zum Einsatz gekommen sind.

5.1.1 Stylesheets

Unter bestimmten Bedingungen bietet es sich an, grafische Formatierungen auszulagern. Dadurch sind diese u. A. für mehrere Komponenten verwendbar und durch die zentrale Ablage leichter zu warten. Für Entwickler, die Flex in Kombination mit PHP nutzen wollen, bietet sich die Möglichkeit, bereits bestehende CSS[1]-Formatierungen übernehmen zu können. Flex unterstützt CSS jedoch nicht vollständig. So sind z. B. Vererbungen oder relative Positionierungen (z. B. "float") noch nicht möglich.

5.1.2 Ladezeiten

Speziell bei Flex-Anwendungen, die große Datenmengen aus einem PHP-Backend beziehen, ist es empfehlenswert, dem Anwender den Ladestatus mitzuteilen. Zwar kann in der Regel keine genaue Voraussage über die Dauer des Vorgangs gemacht werden, jedoch kann eine Rückmeldung (vgl. [Nie94], Heuristik Nr. 5) dem Nutzer die Programmaktivität signalisieren.
Neben der Parametereinstellung `showBusyCursor="true"`, die den Mauszeiger als Uhr darstellt und von mehreren Komponenten (z. B. `<mx:RemoteObject>`) unterstützt wird, kann eine zusätzliche Animation als Hintergrundbild der entsprechenden Komponente eingebunden werden. Dies erweist sich oft als hilfreich, da dem Anwender so mitgeteilt wird, welche Komponente der Applikation sich in Bearbeitung befindet.

[1] Cascading Style Sheets (CSS) ist eine beschreibende Sprache für die Darstellung spezieller Elemente oder Inhalte.

Bei der Animation ist zu beachten, dass das im Internet häufig verwendete “gif”-Format von Flex nicht nativ unterstützt wird. Mit entsprechenden Programmen lassen sich diese jedoch in das “swf”-Format konvertieren und anschließend einbinden.

5.1.3 States und Transitions

In Flex gibt es die Möglichkeit, Zustände zu definieren, so genannte <mx:states>. Diese beschreiben die Gestalt von Komponenten sowie deren relative Anordnung zueinander. Dadurch lässt sich die Programmoberfläche in Abhängigkeit der zu leistenden Aufgabe mit sehr geringem Aufwand anpassen, indem lediglich der gewünschte Zustand gewählt wird. Dieses Konzept wurde für die Login-Funktion des Reportingclients verwendet, welche beim Programmstart sowie bei Ablauf der Session ausgeführt wird. Neben harten Übergängen lassen sich zudem mit Hilfe von <mx:transitions> Übergangseffekte zwischen den einzelnen Sichten definieren. Beim angemessenen Einsatz dieser Effekte kann dadurch auf einfache Weise eine ansprechende Visualisierung erzielt werden.

5.2 Darstellung

In Abbildung 5.1 ist der Flex-basierte Reportingclient mit drei unterschiedlichen Reportingfunktionen dargestellt. Die folgenden Abschnitte behandeln die wesentlichen Bereiche der Applikation und erläutern deren Konzept.

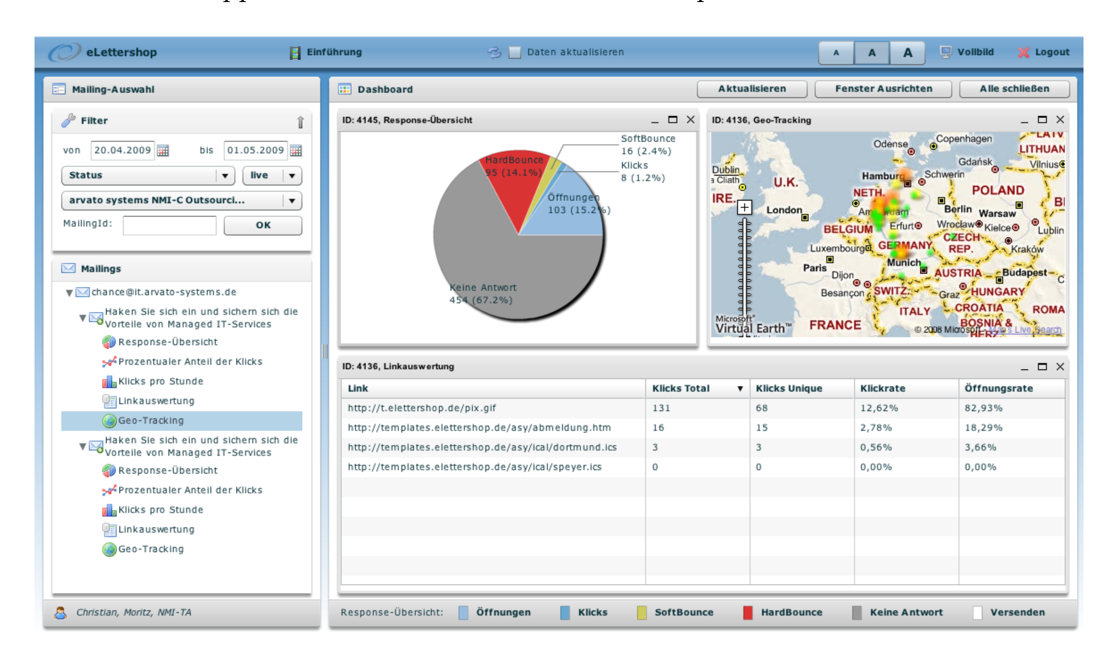


Abbildung 5.1: Der Reportingclient

5.2.1 Menü

Die grundlegenden Optionen für die Steuerung der Applikation lassen sich über die Menüleiste erreichen. Da ihr Funktionsumfang überschaubar ist, erübrigt sich die Verwendung von Drop-Down-Menüs. Die Farbgestaltung lehnt sich an das Corporate Design von arvato an. Daher wird die Menüleiste im arvato-typischen blau abgebildet. Die folgende Auflistung erläutert die einzelnen Funktionen des Menüs.



Abbildung 5.2: Menüleiste

- **Link zum eLettershop**
  Um dem Anwender die Rückkehr vom Reportingclient zum eigentlichen eLettershop zu ermöglichen, wird die Startseite des eLettershops verlinkt. Der Link beinhaltet das Logo von arvato systems und trägt dadurch zur Corporate Identity bei.

- **Einführungsvideo**
  Der Usability-Test hat aufgezeigt, dass das Benutzer-Interface in einigen Bereichen wenig intuitiv ist. So ist z. B. die Existenz der Drag and Drop Funktionalität für viele Anwender nicht direkt ersichtlich, die eigentliche Anwendung der Funktionalität hingegen schon. Diese Unzulänglichkeit wird so behoben, dass der Hinweis auf die Existenz direkt mit einem erklärenden Anleitungsvideo verbunden ist. Unter ergonomischen Gesichtspunkten erscheint diese Umsetzung sinnvoll, da eine hohe Wahrscheinlichkeit besteht, dass der Anwender die Funktion nicht kennt und zudem eine Erklärung abrufen möchte. Hierdurch wird ein fließender Arbeitsablauf gewährleistet.

- **Datenaktualisierung**
  Damit die Empfängerreaktionen auf ein Mailing während des Versands beobachtet werden können, wurde eine automatisch Aktualisierungsfunktion implementiert. Sie sorgt dafür, dass die Darstellung der Reportingdaten in äquidistanten Zeitschritten erneuert wird. Um den Dialog mit dem Benutzer einfach und übersichtlich zu halten, kann der Anwender die Aktualisierungsfrequenz erst einstellen, nachdem die dafür vorgesehende Checkbox aktiviert wurde. Damit die Serverbelastung clientseitig begrenzt wird, beträgt der Aktualisierungsabstand mindestens fünf Sekunden.

- **Schriftgrößenänderung**
  Der Anwender kann in der Applikation zwischen drei Schriftgrößen entscheiden. Die Bedienelemente sind stilisiert durch drei aufeinander folgende “A’s” in aufsteigender Schriftgröße. Diese Symbolik spiegelt auf intuitive Weise die Funktion wider und wird in der Regel auf barrierearmen Webseiten verwendet.

- **Vollbildmodus**
  Obwohl Flash-Dateien in gängigen Webbrowsern auch direkt geöffnet und angezeigt werden können, bietet die Einbettung in HTML weitere Steuerungsmöglichkeiten. So kann im HTML-Template[2] z. B. der Parameter `allowFullscreen` im `object`-Tag so gesetzt werden, dass der Anwender die Flash-Applikation im Vollbildmodus betrachten kann. Um diese Funktion in der Flex-Applikation nutzen zu können, wird ein Bedienelement implementiert, welches folgenden Befehl aufruft:
  `systemManager.stage.displayState=StageDisplayState.FULL_SCREEN;`
  In diesem Modus werden keine Browserelemente dargestellt. Hierdurch steht der Applikation die volle Bildschirmfläche zur Verfügung und kann zudem wie eine Desktop-Applikation wirken.

- **Logout-Funktion**
  Das Ausloggen erfolgt über einen Button, welcher auf die gleiche URL verweist wie die Auslog-Funktion des eLettershops. Da sowohl der Reportingclient als auch der eigentliche eLettershop die gleiche Session verwenden, bietet sich diese Umsetzung an. Der Anwender gelangt beim Logout auf die eLettershop Startseite, die den erfolgreichen Logout mit einem Informationstext quittiert.

5.2.2 Filter

Der Filter stellt die entscheidende Komponente für die Steuerung der Mailingansicht dar. Um den Aspekt der Individualisierbarkeit zu beachten, kann der Anwender das Filterfenster ein- und ausblenden (siehe Pfeil in Abb. 5.3). Beim Eingabefeld der MailingId wird über die Parametereinstellung `restrict="[0-9]"` der Komponente `<mx:TextInput>` die Eingabe von Buchstaben und Sonderzeichen unterbunden. Hierdurch wird dem Usability-Kriterium der Fehlervorbeugung Rechnung getragen. Dieser Aspekt wird auch an anderer Stelle, der Datumseingabe, berücksichtigt. Sie erfolgt über die Terminwahl innerhalb eines implementierten Kalenders. Der Benutzer wird an dieser Stelle von der Einhaltung eines gültigen Datumformates entbunden. Gute Usability zeichnet sich nicht nur durch grundlegende Konzepte aus, sondern lässt sich auch durch eine Vielzahl kleinerer Detaillösungen verbessern. Ein Beispiel hierfür ist die Eingabe-Bestätigung einer MailingId mit Hilfe der Enter-Taste. Die Bedienung ist hierbei ergonomischer als beim Wechsel von der Tastatureingabe zur Maussteuerung.

5.2.3 Mailings

Die Mailings werden wie in Abbildung 5.5 erkennbar als Baumstruktur abgebildet. Der Usability-Test hat bestätigt, dass diese Form der Mailingdarstellung übersichtlich und strukturiert ist. Die Struktur in Form eines Baums wird in Abschnitt 4.1.1 näher erläutert. Zusätzlich zu dieser Aufteilung, werden spezifische Symbole

[2]engl. für Vorlage

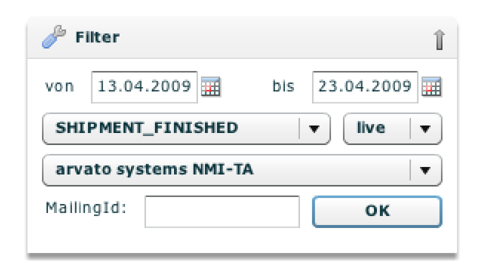


Abbildung 5.3: Filtereinstellungen

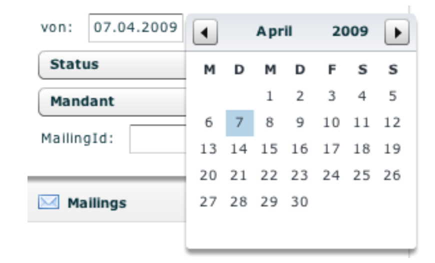


Abbildung 5.4: Datumsauswahl

verwendet. Sie weisen auf die Kategorie bzw. jeweilige Funktion hin. Da gut gewählte Bilder schneller aufgefasst werden können als zu lesender Text, wird die Bedienung hierdurch intuitiver und die Gedächtnislast minimiert. Das Aufklappen der Ebenen ist in dieser Tree-Komponente nativ über ein verhältnismäßig kleines Dreieck-Symbol implementiert. Dieses Bedienelement unterminiert in bestimmten Fällen (z. B. Sehschwäche, Motorikstörung) die Barrierefreiheit. Die ohnehin benötigte Beschriftungsfläche der Hierarchieebene wird daher als Bedienelement zur Öffnung der jeweiligen Knoten verwendet.



Abbildung 5.5: Baumstruktur Mailings

5.2.4 Dashboard

In der Konzeptionsphase wurde das Layout der Reportingdarstellungen mit der Komponente <mx:Tile> geplant. Diese wird jedoch nicht allen Anforderungen (z. B. das individuelle Platzieren einzelner Felder) gerecht. Daher wurde während der Realisierungsphase im Rahmen einer Internetrecherche nach einer geeigneten Bibliothek gesucht. Eine gute technische Lösung stellt hierbei “flexlib”[3] dar. Diese übernimmt das Layout der einzelnen Fenster sowie zusätzliche Funktionen. Hierzu zählen u. A. das Verschieben einzelner Fenster und deren Maximierung resp. Minimierung. Implementiert sind diese Funktionen in den Komponenten <mx:MDICanvas> und <mx:MDIWindow>, welche im Reportingclient Verwendung finden.

[3]siehe `http://code.google.com/p/flexlib`

Häufig benötigte Funktionen können durch Shortcuts ausgeführt werden. So lassen sich z. B. sämtliche angezeigten Reportingfunktionen mittels der Funktionstaste "F5" aktualisieren.
Die Breite der Bereiche Mailing-Auswahl und Dashboard sind durch die Verwendung der Komponente <mx:HDividedBox> skalierbar. Somit kann der Anwender das Dashboard auf die komplette Breite des Browserfensters vergrößern.

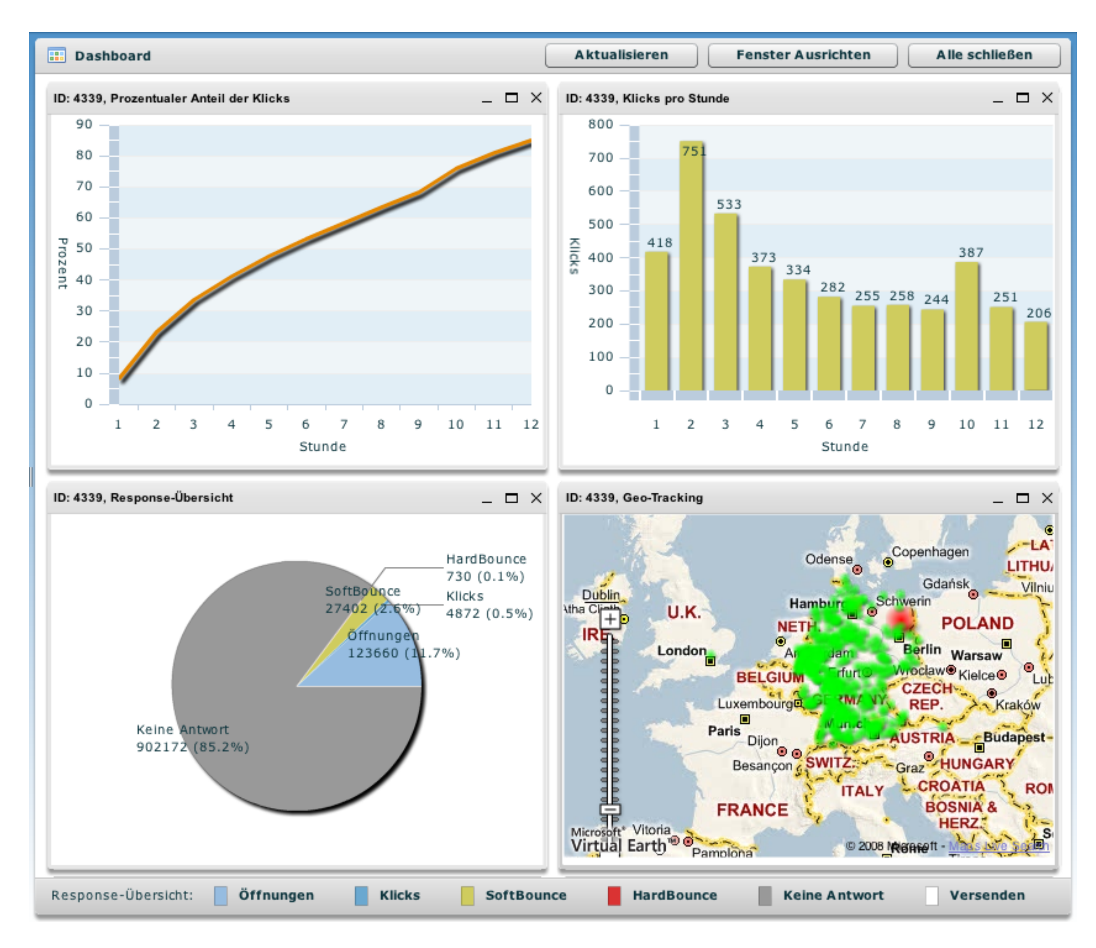


Abbildung 5.6: Dashboard

5.2.5 Reportingfunktionen

Die folgenden Abbildungen stellen Beispiele für Reportingfunktionen dar.

- **Response-Übersicht**
  Die Response-Übersicht wird als Kreisdiagramm visualisiert (vgl. Abb. 5.10), um die prozentuale Verteilung zu verdeutlichen. Die Farben der einzelnen Segmente spiegeln das arvato-Design wider. Die einzige Ausnahme stellt die Kategorie Hard Bounce dar, für welche die Signalfarbe Rot verwendet wird. Der

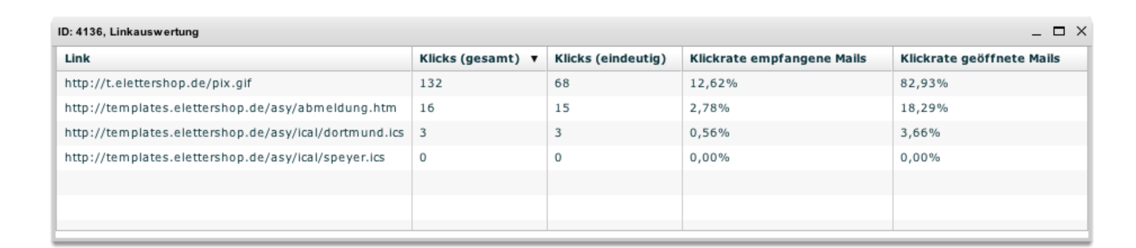
Kunde soll auf diese Weise verstärkt auf ungültige Empfängeradressen hingewiesen werden.

- **Prozentualer Anteil der Klicks**
  Der kumulierte prozentuale Anteil der Klicks pro Stunde wir als zeitlicher Verlauf dargestellt. Dieser lässt sich gut durch ein Liniendiagramm abbilden (vgl. Abb. 5.8), wobei der Graph in eine Sättigung läuft.

- **Klicks pro Stunde**
  Diese Reportingfunktion teilt die insgesamt angefallenen Klicks in Stunden auf. Hierdurch kann der Anwender erkennen, in welcher Stunde nach dem Versand die Klickanzahl am größten war. Die Darstellung als Balkendiagramm (vgl. Abb. 5.9) bietet sich an, da es sich bei den Daten um diskrete Werte handelt.

- **Linkauswertung**
  Flex bietet die Komponente `<mx:DataGrid>` an, um Datensätze in Tabellenform abzubilden. Zusätzlich sind Sortierfunktionen der einzelnen Spalten implementiert. Diese kann der Anwender nutzen, um die im Mailing enthaltenen Links (vgl. Abb. 5.7) nach Klickanzahl zu sortieren. Hierdurch ist eine erste einfache Analyse eines Mailings möglich.

- **Geo-Tracking**
  Das Geo-Tracking zeigt mit Hilfe von Häufungspunkten die geographische Verteilung der Empfänger. Für die Darstellung wurde die digitale Karte "Virtual Earth" verwendet, welche eine KML-Datei mit Geo-Daten abbilden kann (vgl. Abb. 5.11). Im Gegensatz zu "Google Earth"[4] ist diese auch für kommerzielle Zwecke kostenlos verwendbar.



ID: 4136, Linkauswertung

Link	Klicks (gesamt) ▼	Klicks (eindeutig)	Klickrate empfangene Mails	Klickrate geöffnete Mails
http://t.elettershop.de/pix.gif	132	68	12,62%	82,93%
http://templates.elettershop.de/asy/abmeldung.htm	16	15	2,78%	18,29%
http://templates.elettershop.de/asy/ical/dortmund.ics	3	3	0,56%	3,66%
http://templates.elettershop.de/asy/ical/speyer.ics	0	0	0,00%	0,00%

Abbildung 5.7: Auflistung der Links

Neben den einzelnen Reportingdarstellungen, lassen sich ebenfalls mehrere Mailings in einem gemeinsamen Diagramm vergleichen (vgl. Abb. 5.12). Diese Funktion ist lediglich für die Graphendarstellungen sinnvoll. Die hinzugefügten Datensätze können über einen speziellen Button (vgl. Abb. 5.13), der als Titel die ID des Mailings

[4]siehe `http://earth.google.de`

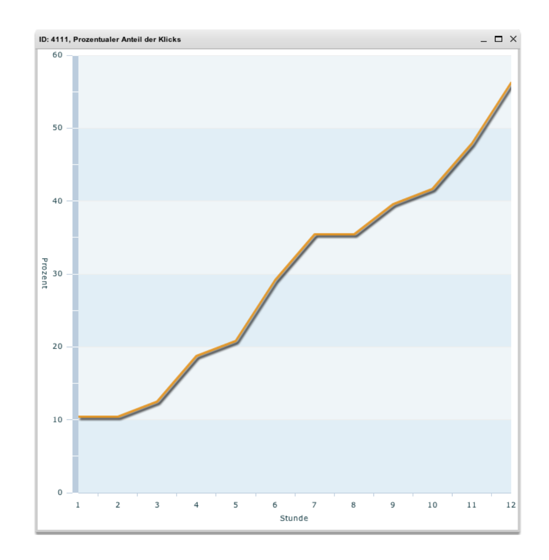


Abbildung 5.8: Liniendiagramm: Anteil der Klicks

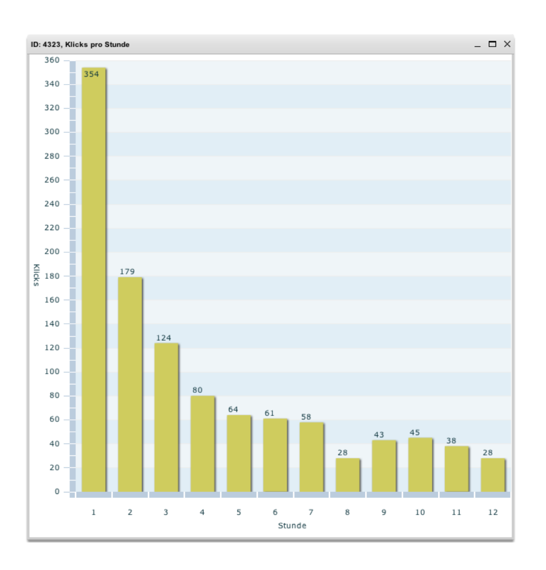


Abbildung 5.9: Balkendiagramm: Klicks pro Stunde

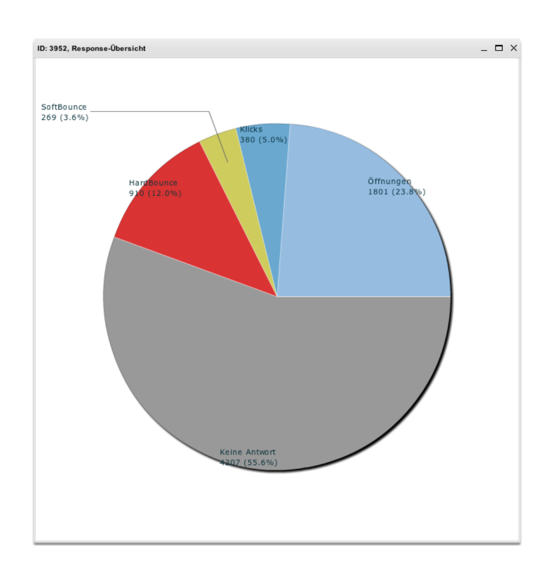


Abbildung 5.10: Kreisdiagramm: Response-Übersicht



Abbildung 5.11: Geo-Tracking: Lokalisierung der Empfänger

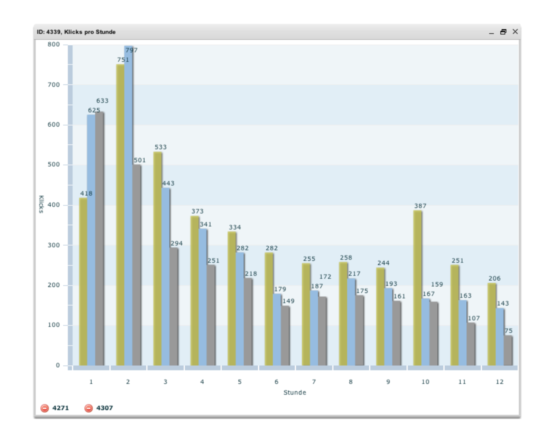


Abbildung 5.12: Balkendiagramm: Mehrere Mailings

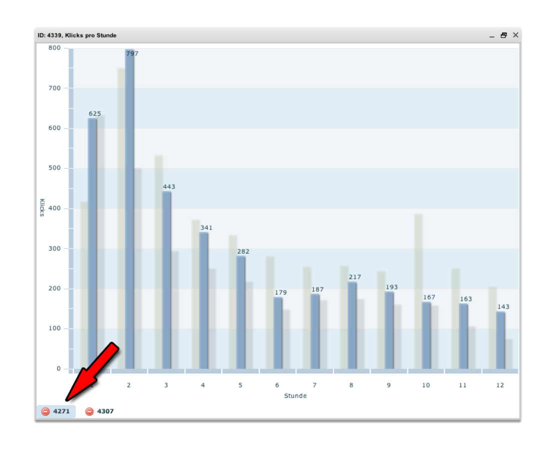


Abbildung 5.13: Steuerungsfunktionen in einem Diagramm

besitzt, wieder entfernt werden. Gleichzeitig dient dieser der Hervorhebung des entprechenden Mailings im Diagramm. Sobald sich die Maus über dem Button befindet, werden die übrigen Mailings transparent dargestellt. Durch die doppelte Funktionsbelegung des Buttons wird eine effiziente Nutzung der Benutzeroberfläche erreicht und die Übersichtlichkeit bleibt erhalten.

5.3 Pitfalls

Um Flex-Entwickler auf mögliche Komplikationen vorzubereiten, sollen in diesem Abschnitt typische Pitfalls[5] erläutert werden. Generelles Problem sind die unpräzisen Fehlermeldungen, besonders bei Verbindungsfehlern über AMF. PHP-Fehler werden generell nicht genau spezifiziert. Der Zend_Amf-Entwickler Wade Arnold merkte an, dass nur dann detaillierte Fehlermeldungen an Flex gesendet werden, wenn das Test- und nicht das Produktivsystem des Zend Frameworks aktiv ist.

> "Otherwise we do not return that data as an exception could have sensitive information that could create a security risk. Especially in the case of a db connection failure."[6]

Für eine geeignete Fehlerdiagnostik bietet sich ein Proxy-Service (z. B. das kostenpflichtige Programm Charles[7]) an, der die Datenübertragung zwischen PHP und Flex protokollieren und anzeigen kann. Dadurch können neben konkreten Problemen (z. B. fehlerhaftes Class-Mapping) auch redundante Datenübertragungen, welche die Performance schwächen, aufgedeckt werden.

[5] engl. für Falle, Fallgrube

[6] vgl. `http://corlan.org/2009/01/07/throwing-an-error-when-working-with-php-and-amf`, zugegriffen am 29. April 2009

[7] siehe `http://www.charlesproxy.com`

5.3.1 Mehrsprachigkeit

Bei Erstellung der Sprachdateien ist auf das Dateiformat zu achten. Hier empfiehlt sich UTF-8, um auch Sonderzeichen darstellen zu können. Der Flex Builder gibt als Standard das Format ISO-8859-1 vor. Bei der Verwendung des Standardformats wurden in diesem Projekt nach dem Einfügen der Sprachdaten sämtliche Sonderzeichen fehlerhaft dargestellt.

5.3.2 Sessions

Es existieren zwei Möglichkeiten, die Verbindung von Flex zu einem PHP-Backend mit Zend Framework herzustellen. Die am häufigsten im Netz dokumentierte Variante ist die Verwendung einer PHP-Datei als Endpoint. Dabei wird jedoch lediglich der Zend_Amf Server initialisiert. Das Session-Handling muss in diesem Fall manuell erfolgen.
Die günstigere Variante stellt daher die Initialisierung in einem Controller (vgl. Abs. 2.2) dar. Dabei wird die Verwaltung der Sessions von dem Zend Framework übernommen. Grundsätzlich läuft zwar jede Anfrage von Flex über den Endpoint, ist dieser jedoch im Controller lokalisiert, so werden die im Zend Framework implementierten Standard-Befehle (u. A. für das Session-Handling) automatisch ausgeführt.

5.3.3 Remote-Objekte

Der Entwickler sollte sich im Klaren darüber sein, dass PHP keinen ausführbaren Code an Flex senden kann. Somit beschränkt sich die Kommunikation zwischen Flex und PHP-Backend auf den Austausch von Daten. Im Abschnitt 3.5 wird der Umgang mit dieser Restriktion näher erläutert.

6 Zusammenfassung und Ausblick

Im Folgenden wird zunächst auf die Weiterentwicklung des Reportingclients eingegangen. Anschließend wird die Bewertung speziell von Flex sowie allgemein von RIAs aufgrund der gemachten Erfahrungen und des gesammelten Wissens vorgenommen. Den Abschluss bildet der Ausblick auf die sich abzeichnende Entwicklung von Webanwendungen.

Der Reportingclient konnte ausnahmslos um die vorgesehenen Funktionen erweitert werden. Das Feedback der Probanden hinsichtlich des Designs, der Performance sowie der Bedienung ist als überaus positiv zu bewerten. Anhand dieses konkreten Anwendungsfalls wurde gezeigt, dass die Realisierung von grafiklastigen sowie interaktionsreichen Webanwendungen mittels Flex gut geeignet ist. Ein Nachteil dieser Technologie ist das Fehlen einer nativen Datenbankunterstützung. Um diese Unzulänglichkeit zu beheben, bedarf es einer serverseitigen Lösung. Ihre technische Umsetzung gestaltete sich aufwendiger als ursprünglich angenommen, da hierzu das PHP-Backend um zusätzliche Klassen (VOs) erweitert werden musste.

Bei der richtigen technischen Umsetzung einer Flex-Applikation sind in den Bereichen Plattformunabhängigkeit, Usability, Barrierefreiheit sowie Sicherheit und Performance keine nennenswerten Probleme zu erwarten. Die Akzeptanz der Anwender gegenüber dieser Technologie wurde anhand einer empirischen Evaluierung gezeigt.

Die Verwendung von RIAs ist grundsätzlich für jede Anwendung möglich, jedoch nicht immer sinnvoll. Im Falle von schlichtgehaltenen Webseiten ist die Verwendung von RIAs tendenziell überdimensioniert. Allerdings gilt, dass komplexere Webangebote (z. B. Internet-Shops, Groupware[1]) komplexere Technologien zur effizienten Umsetzung erfordern.

Allgemein geht der Trend hin zu desktopähnlichen und somit komplexeren Webanwendungen. Joseph Reger[2] wagt bezüglich der Betriebssystementwicklung folgende Zukunftsprognose:

> "Der Browser wird in diesem Umfeld zum Client sowie zum zentralen Programm, das User bei der Arbeit am Computer benutzen. Das Betriebssystem selbst rückt dabei zunehmend in den Hintergrund."[3]

[1] Bei Groupware handelt es sich um kollaborative Webapplikationen, wie z. B. Google Docs (siehe `http://docs.google.com`).

[2] Joseph Reger ist Chief Technology Officer (CTO) von Fujitsu Siemens Computers.

[3] vgl. `http://pressetext.de/news/081113019/pc-zukunft-browser-loest-betriebssystem-ab`, zugegriffen am 3. Mai 2009

Wie weit diese Entwicklung bereits fortgeschritten ist, zeigt die spezielle Linux Distribution Splashtop. Dabei handelt es sich um ein äußerst schlankes eingebettetes System, welches neben einer grafischen Oberfläche lediglich einen Webbrowser und ein VoIP[4]-Programm zur Verfügung stellt. Verfolgt man diesen Gedanken weiter, so reduziert sich die Funktion des Betriebssystems auf die Ansteuerung der Rechner-Hardware, so dass die Funktionalität herkömmlicher Betriebssysteme durch im Browser ausgeführte Applikationen übernommen wird. Im Falle einer solchen Entwicklung stellen RIAs einen wesentlichen Bestandteil der zukünftigen Informationstechnik dar.

[4]"Voice over IP" (VoIP) ermöglicht das Telefonieren über Computernetzwerke.

Anhang

A eLettershop Oberfläche

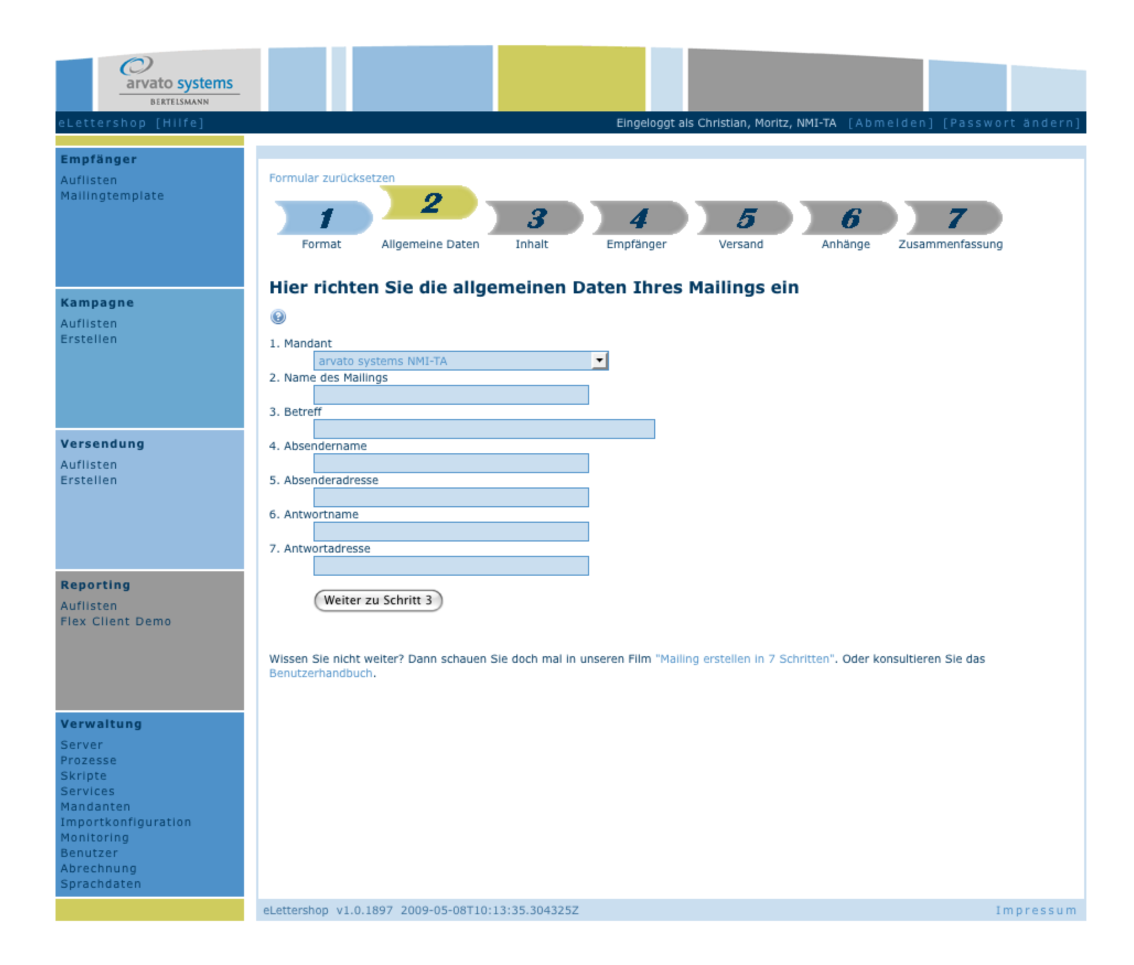


Abbildung A.1: Mailing erstellen

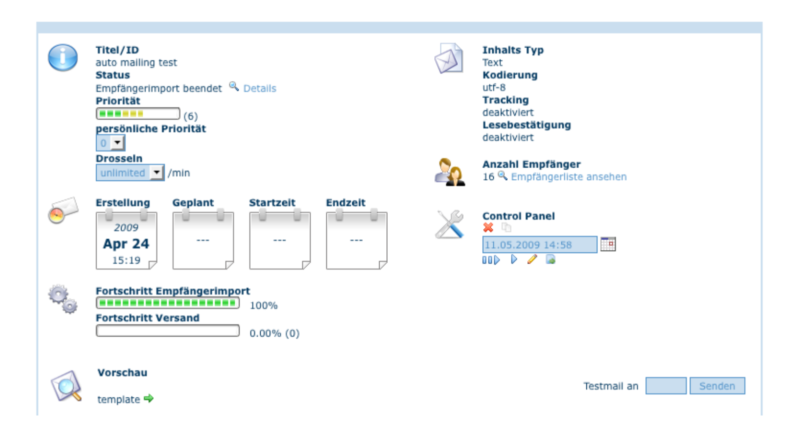


Abbildung A.2: Mailing ansehen



Mailingdaten

Absender	arvato systems (info-dortmund@arvato-systems.de)
Betreff	Reminder: Einladung zum IT_breakfast am 15. Mai 2009
Titel/ID	Reminder: Einladung IT Breakfast 15.05.2009
Ben. def. Feld 1	
Ben. def. Feld 2	
Erstellung	Mo., 04.05.2009 18:38
Geplant	Mi., 06.05.2009 14:00
Startzeit	Mi., 06.05.2009 14:00
Endzeit	Mi., 06.05.2009 14:01
Dauer	00:01:42
Geschwindigkeit	51.494 Empfänger pro Stunde

Responseübersicht

Brutto Versandmenge	1.462	
Anzahl Hard-Bounces	57	3,90%
Anzahl Soft-Bounces	21	1,44%
Anzahl Bounces gesamt	78	5,34%
Anzahl empfangener Mails (Netto Versandmenge)	1.384	94,66%

Abbildung A.3: Reporting ansehen

B CD Inhaltsverzeichnis

```
Demo/  //Vorstellung des Programms
    Video_Bedienung.avi
    Video_Bedienung.mov
Dokumentation/
    LaTeX/  //LaTeX Quelldateien
    ChristianDA.pdf  //Die eigentliche Dokumentation
Elektronische_Literatur/
    AMF3-Spezifikation.pdf
    eLettershop_Benutzerhandbuch.pdf
    ISO_9241-10.pdf
    MihaiCorlan_Presentation_Flex_AIR_and_PHP.pdf
    NielsonNormanGroup_RIA-usability.pdf
    Verwendete_URLs.html
Flex/
    release/  //Die kompilierte Version mit HTML-Einbettung
    src/ //Quellcode
    FlexProjekt.zip  //Flex-Projekt-Datei
PHP/  //Quelldateien aus dem PHP-Backend
    MailingVO/
        MailingRemoteVO.php
        Remote.php
    ReportingVO/
        Remote.php
        ReportingRemoteVO.php
    UserVO/
        Remote.php
        UserRemote.php
    initFlex.php
LIESMICH.TXT
```

Abbildungsverzeichnis

1.1 Reporting im Regelkreis 2
1.2 Internationaler MTV-Webauftritt 1997 4
1.3 Deutscher MTV-Webauftritt 2009 4

2.1 Model-View-Controller Konzept 8
2.2 Flex Architektur 9
2.3 Die Übermittlung einer E-Mail 12
2.4 Tracking-Ablauf 15

3.1 Silverlight Architektur 22
3.2 JavaFX Plattform 23
3.3 RIA-Vergleich 25
3.4 Integration von Flex und PHP 26
3.5 Ablauf eines RPCs 28
3.6 Aufbau einer SOAP-Nachricht 31
3.7 RIA Data Benchmark 33
3.8 Class Mapping 35

4.1 Projektskizzierung 38
4.2 Baumstruktur 39
4.3 Combobox und Liste 40
4.4 Ablauf der Filterung 41
4.5 Kreisdiagramm 42
4.6 Baumstruktur mit Reportingfunktionen 44
4.7 Baumstruktur: Drag and Drop 44
4.8 Aktivitätendiagramm: Mailingvergleich 48
4.9 ValueObject erhalten 49
4.10 Remote-Objekt in Flex erstellen 51
4.11 CHAP: 3-Wege-Handshake 54
4.12 Papier-Prototyp 56
4.13 Layout der Tile-Komponente 57
4.14 Überprüfung auf W3C-Standardkonformität 58
4.15 Usability-Test: Mündlicher Vortrag 65
4.16 Usability-Test: Aufgabenstellung 66
4.17 Usability-Test: Mailings 66

5.1 Der Reportingclient 70
5.2 Menüleiste 71
5.3 Filtereinstellungen 73
5.4 Datumsauswahl 73
5.5 Baumstruktur Mailings 73
5.6 Dashboard 74
5.7 Auflistung der Links 75
5.8 Liniendiagramm: Anteil der Klicks 76
5.9 Balkendiagramm: Klicks pro Stunde 76
5.10 Kreisdiagramm: Response-Übersicht 76
5.11 Geo-Tracking: Lokalisierung der Empfänger 76
5.12 Balkendiagramm: Mehrere Mailings 77
5.13 Steuerungsfunktionen in einem Diagramm 77

A.1 Mailing erstellen 81
A.2 Mailing ansehen 82
A.3 Reporting ansehen 82

Quellcodeverzeichnis

2.1 Beispiel MXML-Komponente 10
2.2 Beispiel ActionScript-Funktion 10
2.3 Beispiel Flex Kreisdiagramm 11
3.1 Beispiel Silverlight-Button 22
3.2 Beispiel JavaFX-Button 24
3.3 Beispiel HTTPService Objekt 30
4.1 Zend_Amf_Server 50
4.2 Beispiel Class Mapping UserVO 50
4.3 Beispiel RemoteObject 51
4.4 Beispiel AsyncToken und ItemResponder 52
4.5 Tile Layout 57

Literaturverzeichnis

[ALB08] Rob Allen, Rick Lo, and Stephen Brown. *Zend Framework in Action.* Manning Pubn, 2008.

[BG08] Andreas Bauer and Holger Günzel. *Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung.* dpunkt Verlag, 2008.

[Hel05] Jan Eric Hellbusch. *Barrierefreies Webdesign? Praxishandbuch für Webgestaltung und grafische Programmoberflächen.* dpunkt.verlag, 2005.

[KW02] Michael Kuschke and Ludger Wölfel. *Web Services kompakt.* Spektrum Akadademischer Verlag, 2002.

[Moo07] Colin Moock. *Essential ActionScript 3.0.* O'Reilly Media, 2007.

[MS04] Christoph Meine and Harald Sack. *WWW: Kommunikation, Internetworking, Web-Technologien.* Springer, 2004.

[NA08] Joshua Noble and Todd A. Anderson. *Flex 3 Cookbook.* O'Reilly Media, 2008.

[Nie94] Jakob Nielsen. *Usability Engineering.* Academic Press International, 1994.

[Pit86] David H. Pitt. *Category Theory and Computer Programming.* Springer, 1986.

[RR07] Leonard Richardson and Sam Ruby. *Web Services mit REST.* O'Reilly, 2007.

[Szw08] Gerd Szwillus. *Usability Engineering. Vorlesung.* Universität Paderborn, Fakultät für Elektrotechnik, Informatik und Mathematik, Paderborn. 2008.

[Wid08] Simon Widjaja. *Rich Internet Applications mit Adobe Flex 3.* Hanser Fachbuch, 2008.